НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях»

Виконав:
студент IV курсу, групи КП-61
Ільїн Максим Олександрович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.т.н., доцент,
Дідковська Марина Віталіївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

«ЗАТВЕРДЖУЮ»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Ільїну Максиму Олександровичу

1. Тема проєкту «Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях», керівник проєкту Заболотня Тетяна Миколаївна, к.т.н., доцент, затверджені наказом по університету від «25» травня 2020 р. №1181-с

2. Термін подання студентом проєкту «12» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз мов програмування та технологій розроблення;
   - розроблення алгоритмів роботи програми;
   - розроблення дизайну сторінок та графічних елементів;
   - опис програмної реалізації системи;
   - аналіз розробленого веб-додатку.

5. Перелік обов'язкового графічного матеріалу:
   - загальна архітектура системи (креслення);
   - схема потоку алгоритму навчання класифікатора (креслення);
   - діаграма прецедентів веб-додатку (плакат);
   - схема потоку навчання моделі веб-додатку (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання: «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	18.10.2019	
2.	Розроблення та узгодження технічного завдання	21.11.2019	
3.	Розроблення структури web-ресурсу	07.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	18.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	01.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	28.02.2020	
7.	Програмна реалізація web-ресурсу	25.03.2020	
8.	Тестування web-ресурсу	10.04.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	03.05.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	21.05.2020	
11.	Підготовка графічної частини дипломного проєкту	03.06.2020	
12.	Оформлення документації дипломного проєкту	06.06.2020	

Студент Максим ІЛЬЇН

Керівник проєкту Тетяна ЗАБОЛОТНЯ

АНОТАЦІЯ

Дана дипломна робота присвячена розробці програмного забезпечення, що напрямлене на класифікацію новин, а саме присвоєння новині статусу «True» або «Fake» в результаті лексичного аналізу її тексту методами машинного навчання.

В роботі виконано порівняльний аналіз уже існуючих програмних та алгоритмічних рішень для виконання аналізу текстових новин, обґрунтовано вибір системи управління базами даних, мов програмування, що використані для написання клієнтської та серверної частини системи. Розглянуто різні шляхи реалізації клієнтської частини додатку і аргументовано кінцевий вибір.

У даному дипломному проєкті запропоновано алгоритм класифікації текстових новин, що базується на послідовній моделі із чотирма шарами. Розроблено розширення до веб-браузера Google Chrome, що виконує класифікацію новин у мережі інтернет та надає рейтинг сайту-джерелу. Описано розроблену структурно-алгоритмічну реалізацію програмного рішення з точки зору алгоритму класифікації, архітектурної організації бази даних та візуальної репрезентації результату аналізу.

ABSTRACT

This thesis is devoted to the development of software aimed at classifying news, namely the assignment of news status «True» або «Fake» as a result of lexical analysis of its text using machine learning algorithms.

The paper contains a comparative analysis of existing software and algorithmic solutions to analyze text news, substantiates the choice of database management system, programming languages used to write the client and server part of the system. Various ways of realization of a client part of the application are considered and argumentation of a final choice is argued.

An algorithm for classifying text news based on a Sequential model with four layers is proposed. An extension, which classifies news on the Internet and provides a ranking of the source site, has been developed for the Google Chrome web browser. The developed structural-algorithmic implementation of the software solution for the classification algorithm, architectural organization of the database and visual representation of the analysis result is described.

АННОТАЦИЯ

Данная дипломная работа посвящена разработке программного обеспечения, направленного на классификацию новостей, а именно присвоение новости статуса «True» або «Fake» в результате лексического анализа ее текста методами машинного обучения.

В работе выполнен сравнительный анализ уже существующих программных и алгоритмических решений для выполнения анализа текстовых новостей, обоснован выбор системы управления базами данных, языков программирования, используемых для написания клиентской и серверной части системы. Рассмотрены различные пути реализации клиентской части приложения и приведена аргументация конечного выбора.

В данном дипломном проєкте предложен алгоритм классификации текстовых новостей, основанный на последовательной модели с четырьмя слоями. Разработано расширение для веб-браузера Google Chrome, которое выполняет классификацию новостей в сети интернет и предоставляет рейтинг сайта-источника. Описана разработанная структурно-алгоритмическая реализация программного решения с точки зрения алгоритма классификации, архитектурной организации базы данных и визуальной репрезентации результата анализа.

ДП.045440-01-90 Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Веб-додаток для	4	
	автоматизованого		
	виявлення фіктивних		
	текстових новин у		
	Інтернет-виданнях.		
	Технічне завдання		
ДП.045440-03-81	Веб-додаток для	69	
	автоматизованого		
	виявлення фіктивних		
	текстових новин у		
	Інтернет-виданнях.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-додаток для	4	
	автоматизованого		
	виявлення фіктивних		
	текстових новин у		
	Інтернет-виданнях.		
	Програма та методика		
	тестування		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-05-34	Веб-додаток для	5	
	автоматизованого		
	виявлення фіктивних		
	текстових новин у		
	Інтернет-виданнях.		
	Керівництво		
	користувача		
ДП.045440-06-99	Веб-додаток для	1	
	автоматизованого		
	виявлення фіктивних		
	текстових новин у		
	Інтернет-виданнях.		
	Загальна структура		
	системи. UML діаграма		
ДП.045440-07-99	Веб-додаток для	1	
	автоматизованого		
	виявлення фіктивних		
	текстових новин у		
	Інтернет-виданнях.		
	Навчання класифікатора.		
	Схема алгоритму		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-08-98	Веб-додаток для	1	
	автоматизованого		
	виявлення фіктивних		
	текстових новин у		
	Інтернет-виданнях.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«__» _____ 2019 р.

ВЕБ-ДОДАТОК ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ФІКТИВНИХ ТЕКСТОВИХ НОВИН У ІНТЕРНЕТ-ВИДАННЯХ

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Максим ІЛЬЇН

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання у веб-браузерах в якості сервісу для класифікації новин, що надаються користувачем, як правдивих або фіктивних.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмний продукт повинен забезпечувати такі основні функції:

1) можливість використання у веб-браузері;

2) можливість аналізу довільного об'єму тексту новини з веб-сторінки;

3) можливість отримання рейтингу сайту-джерела новини.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

– «Загальна архітектура системи. UML-діаграма»;
– «Схема потоку алгоритму навчання класифікатора».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою проєкту 18.10.2019
Розроблення та узгодження технічного завдання 21.11.2019
Розроблення архітектури програмного забезпечення 07.12.2019
Розроблення дизайну інтерфейсів 01.02.2020
Програмна реалізація дипломного проєкту 25.03.2020
Тестування реалізованої системи 10.04.2020
Підготовка матеріалів текстової частини проєкту 03.05.2020
Підготовка матеріалів графічної частини проєкту 25.05.2020
Оформлення технічної документації проєкту 03.06.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ВЕБ-ДОДАТОК ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ФІКТИВНИХ ТЕКСТОВИХ НОВИН У ІНТЕРНЕТ-ВИДАННЯХ

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Максим ІЛЬЇН

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ОС – операційна система.

ПЗ – програмне забезпечення.

Веб-браузер – програмне забезпечення для перегляду сторінок та файлів в мережі Інтернет.

Машинне навчання – клас методів штучного інтелекту, характерною рисою яких є навчання в процесі застосування рішень безлічі подібних завдань.

Персептрон – математична або комп'ютерна модель сприйняття інформації мозком.

MVC – схема поділу даних програми, користувацького інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Автентифікація – перевірка користувача шляхом порівняння введеного їм пароля (для зазначеного логіна) з паролем, збереженим в базі даних користувацьких логінів.

Кросплатформність – здатність програмного забезпечення працювати з двома і більше апаратними платформами і операційними системами.

Парадигма програмування – система правил, що визначає стиль написання програми.

JRE – мінімальна реалізація віртуальної машини, необхідна для виконання Java-додатків, без компілятора і інших засобів розробки.

Розширення браузера – спеціальне програмне забезпечення для веб-браузера, основне призначення якого – розширення функціоналу браузера.

HTML – стандартизована мова розмітки гіпертексту, що використовується для створення документів в мережі Інтернет.

CSS – формальна мова, що використовується для опису зовнішнього вигляду веб-сторінки.

Фреймворк – набір абстракцій, які надають базову функціональність програмного забезпечення для подальшої композиції та модифікації.

Скрипт – короткий опис дій, що виконується системою.

БД – база даних.

СКБД – система управління базами даних, що забезпечує інтерфейс для створення бази даних та її модифікації.

API – прикладний програмний інтерфейс, що надає набір рішень, за допомого яких програма може взаємодіяти з іншою програмою.

REST – архітектурний стиль взаємодії компонентів розподіленого додатку в мережі Інтернет.

Чатбот – програмне забезпечення, що використовується для ведення онлайн розмови в чаті за допомогою тексту, замість прямого контакту з опонентом-людиною.

Датасет – певний набір даних, що використовується для тренування та тестування моделі машинного навчання.

Стоп-слова – це слова, які не несуть смислового навантаження і можуть бути видаленні для оптимізації процесу аналізу тексту.

ВСТУП

З кожним днем суспільство все більше починає розуміти цінність інформації, що залишає та споживає в мережі Інтернет. В 2014 році оприлюднені Едвардом Сноуденом дані про шпигунську діяльність США показали масштаб інформації, що збирається, а дещо пізніше скандал із Cambridge Analytica наглядно продемонстрував шляхи і наслідки її використання: демонстрація персоналізованої реклами на базі портрету користувача з метою маніпуляції його рішеннями. Дана історія отримала таке широке розповсюдження не в останню чергу через те, що спільнота почала відкривати для себе крок за кроком, як новітні технології можуть використовуватися з метою маніпуляції та дезінформації. Менш складним за збір і аналіз великих даних, та не менш ефективним способом для реалізації описаних задач є генерування та розповсюдження фіктивних новин – «інформаційна містифікація або навмисне поширення дезінформації в соціальних медіа і традиційних ЗМІ з метою введення в оману, для отримання фінансової або політичної вигоди» [1].

На даний момент 4.57 млрд [2] чоловік мають пристрої із доступом до Інтернету, що робить їх потенційними читачами текстових Інтернет-медіа. Даний факт також підтверджує статистика по споживанню новин: онлайн джерела вже не перший рік переважають над друкованими виданнями. В онлайн виданнях термін “Fake News” отримав поширення в 2016 році [3] і з того часу не покидав суспільного інформаційного середовища.

Фіктивні новини породжують широкий спектр проблем: від слідування читачами фіктивним медичним рекомендаціям до суспільної паніки, дестабілізації політичних ситуацій. За стилем написання і джерелами розповсюдження фіктивні новини, з точки зору читача, важко відрізнити від правдивих, а використання ручної класифікації не може слугувати рішенням проблеми через обсяг безперервно генеруємих новин. Автоматизовані рішення-аналоги спрямовані на реалізацію алгоритму

класифікації, ігноручи практичне використання системи кінцевим користувачем. Саме тому створення автоматизованої системи із забезпеченням інтуїтивно зрозумілого користувацького інтерфейсу для вирішення даної проблеми, постає актуально задачею.

Таким чином, даний дипломний проєкт присвячено розробленню програмного забезпечення, метою якого є визначення ступені правдивості текстової новини із забезпеченням значення показника точності вище 90%.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Суть проблеми фіктивних новин

Термін «фіктивні новини» ввійшов у наш лексикон після виборів президента США 2016 року. Саме в тому році Collins Dictionary присвоїв даному терміну звання «слово року», оскільки його присутність в полі суспільного дискурсу збільшилася на 365% за 12 місяців [4].

Фіктивні новини – це маніпулятивна видозміна реальних фактів, їх спотворення, мета якого може відрізнятися [5]. Проте, незалежно від мети, фіктивні новини характеризуються негативними наслідками.

На виявлення фіктивних новин спрямована діяльність як великих корпорацій [6], так і окремо взятих активістів і суспільних організацій.

Проблема полягає у відсутності абсолютно автоматизованих засобів для виявлення недостовірної інформації. Найбільш точні рішення використовують людський ресурс для класифікації новин, що є не ефективним підходом. Також проблемою даного підходу вирішення проблеми є щоденне збільшення кількості неправдивих новин у мережі Інтернет. Час, що необхідний для обробки такого обсягу інформації при мануальній класифікації новин достатньо великий, що сприяє широкому поширенню неправдивої інформації до того, як вона буде спростована.

Великі компанії також пропонують свої рішення даної проблеми. На початку цього року WhatsApp поділився статистикою блокувань акаунтів з метою запобігання поширенню фіктивних новин: щомісяця в месенджері блокується 2 мільйони акаунтів, шляхом перегляду підозрілих аккаунтів і повідомлень спеціальною групою людей. Також компанія почала роботу над своїм алгоритмом машинного навчання для автоматизації вирішення проблеми [7].

З вищенаведеного бачимо, що задля скорочення комплексу негативних наслідків екстенсивних підходів до класифікації текстових новин, спільнотою експертів з машинного навчання було розпочато роботу

над розробленням автоматизованого детектора фіктивних новин. Основна задача, що наразі стоїть перед спільнотою розробників полягає у тому, щоб дослідити, як технології штучного інтелекту, зокрема машинне навчання та обробка природної мови, можуть бути використані для боротьби з проблемою підроблених новин. Дані технології є перспективними з точки зору визначення правдивості історії або окремих її фактів.

1.2. Огляд популярних методик виявлення фіктивних новин

Розглянемо існуючі методики, що використовують засоби машинного навчання для вирішення наведеної проблеми.

1.2.1. Аналіз заголовків новини

Даний метод полягає у формуванні певної бази заголовків, класифікованих за правдивістю, виявлення певних паттернів у заголовків певних категорій і подальшим навчанням математичної моделі на їх основі. Негативною стороною даної методики можна вважати важкість класифікації усієї статті по заголовку, оскільки заголовок може використовувати цитати, бути сатирично забарвленим.

1.2.2. Аналіз тексту новини

Даний метод дуже схожий до попереднього, але є більш точним за рахунок збільшення об’єму тексту для аналізу. Проте негативна сторона, що стосується ситуацій, коли вся стаття є сатиричною, залишається незмінною і для даного підходу.

В основному даний тип аналізу зосереджується на різних особливостях тексту, а потім включенні цих ознак у класифікаційні моделі, наприклад у Метод опорних векторів, Дерево рішень, Логістичну регресію і т.д. Після оцінки результатів обирається модель із найбільшою точністю.

1.2.3. Виявлення позиції

Реалізація даного методу може варіюватися, тому наведемо його суть. Виявлення позиції передбачає оцінку точки зору двох фрагментів тексту відносно теми, що піднята у новині. Цей підхід може використовувати

емоційний аналіз блоку тексту, або співставлення тексту новини із заголовком. У останньому випадку текст новини може підтверджувати заголовок, суперечити йому, розкривати його тему, або взагалі не мати відношення до заголовку. На базі зроблених висновків робиться гіпотеза про правдивість або фіктивність новини.

1.2.4. Аналіз метаданих новини

Численні досліди [8, 9, 10] доводять існування певних паттернів у мета-інформації новин, що були класифіковані, як фіктивні. Для аналізу можуть використовуватися такі характеристики, як час публікації, місцезнаходження та ім’я автора тощо.

Як і у наведених вище підходах, в результаті виявлення такого паттерну і навчання на його основі математичної моделі, можливе створення автоматизованого інструменту класифікації певної точності.

1.2.5. Перевірка фактів

Даний підхід є буквальною автоматизацією людської діяльності з ручної класифікації новин шляхом перевірки правдивості фактів, приведених у новині. Він зводиться до мануального пошуку підтвердження, або спростування фактів, що наведені у новині у спеціальній власній базі даних або безпосередньо у мережі Інтернет. Попри високу точність, негативною стороною даного підходу є використання великого обсягу ресурсів, недоступних пересічному розробнику.

Розглянувши існуючі методи та техніки аналізу новин для класифікації їх як правдивих або фіктивних, було прийняте рішення в рамках даного дипломного проєкту сфокусуватися саме на дослідженні алгоритму лексичного аналізу тексту новини, оскільки даний підхід забезпечує доволі високу точність класифікації, що може сягати 99% [11], та підходить під апаратні ресурси, які виділені під розроблення.

1.3. Аналіз існуючих програмних рішень

1.3.1. Talos Intelligence

Talos Intelligence – відкрите програмне забезпечення, поширюване за ліцензією Apache 2.0 і створене в рамках Fake News Challenge Stage 1: Stance Detection [12].

Після успішної реалізації декількох методів, командою було виявлено, що найвища точність класифікації досягається використанням ансамблевого навчання, що полягає в поєднанні двох методів: остаточним рішення команди став комбінований алгоритм, що базувався на середньозваженому між деревами рішень, підсиленими градієнтами, та глибокою згортковою нейронною мережею.

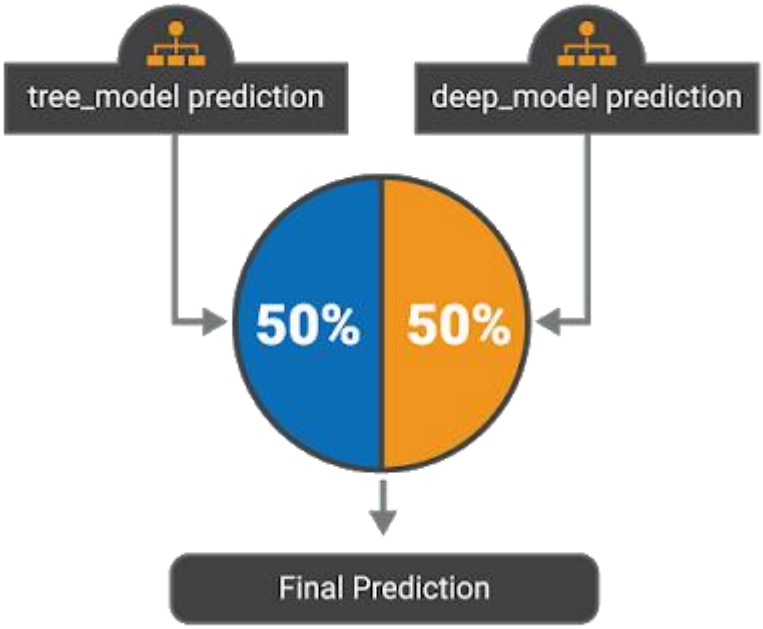


Рис. 1.1. Схематична ілюстрація побудови фінального передбачення Talos Intelligence [13]

За результатами тестів, даний програмний продукт зайняв перше місце у змаганні із Рахунком Відповідності Істині – 82.02 % [13].

Функціональними особливостями даного ПЗ є:

- доволі висока точність виявлення неправдивих новин (82.02 %);

- використання комбінованого підходу шляхом передбачення вірогідності фіктивності заголовку новини деревом рішень та передбачення вірогідності фіктивності тексту новини глибокою згортковою нейронною мережею.

Недоліками даного програмного продукту можна вважати:

- відсутність графічного інтерфейсу користувача;
- відсутність можливості апробації програмного продукту без програмістського бекграунду, за рахунок необхідності розуміння роботи двох модулів із математичними моделями для налаштування ПЗ;
- точність виявлення, хоча і перевищує 80%, але не дозволяє вважати розробку абсолютним рішенням проблеми виявлення фіктивних новин.

1.3.2. athene_system

Команда Athene посіла друге місце у змаганнях Fake News Challenge Stage 1: Stance Detection.

Дане рішення використовує при побудові математичної моделі багатошаровий персептрон (MLP). Автори розширюють оригінальну структуру моделі на шість прихованих softmax-шарів, що містять кілька функцій, спрямованих на лексичний аналіз текстів.

В залежності від типу вибраної функції, користувач на виході отримує один із трьох варіантів:

- вектор ознак заголовка;
- вектор ознак тексту;
- комбінований вектор ознак тексту і його заголовка.

За результатами тестів, даний програмний продукт зайняв друге місце у змаганні із Рахунком Відповідності Істині – 81.97 % [14].

Функціональними особливостями даного ПЗ є:

- доволі висока точність виявлення неправдивих новин (81.97 %);
- використання декількох функцій лексичного аналізу, що надає можливість їх комбінування при застосуванні окремо до заголовків і тексту статей;
- використання багатошарового персептрону для побудови математичної моделі.

Недоліками даного програмного продукту можна вважати:

- відсутність графічного інтерфейсу користувача;
- відсутність можливості апробації програмного продукту без програмістського бекграунду;
- наявність особливих вимог до системи, на якій буде проводитися запуск ПЗ. До таких вимог відноситься версія мови програмування python, спеціальні версії програмних бібліотек;
- для забезпечення заявленої точності необхідно мати 0.9.0 версію бібліотеки tensorflow у режимі GPU;
- фінальні результати представлені у вигляді .csv файла, візуалізація відсутня;
- необхідність встановлення шести спеціальних бібліотек із різних джерел;
- необхідність параллельного запуску Stanford-парсеру;
- точність виявлення хоча і є порівняно високою, але не є абсолютним рішенням проблеми виявлення фіктивних новин.

1.3.3. UCL Machine Reading

UCL Machine Reading – проєкт, що посів третє місце у змаганнях Fake News Challenge Stage 1 (Fnc-I): Stance Detection.

Рішення ґрунтується на єдиній цілісній системі, яка складається з лексичних та схожих ознак, що передаються через багатошаровий персептрон з одним прихованим шаром [15].

Хоча порівняно проста за своєю суттю, система працює нарівні з більш досконалими, ансамблевими системами інших колективів.

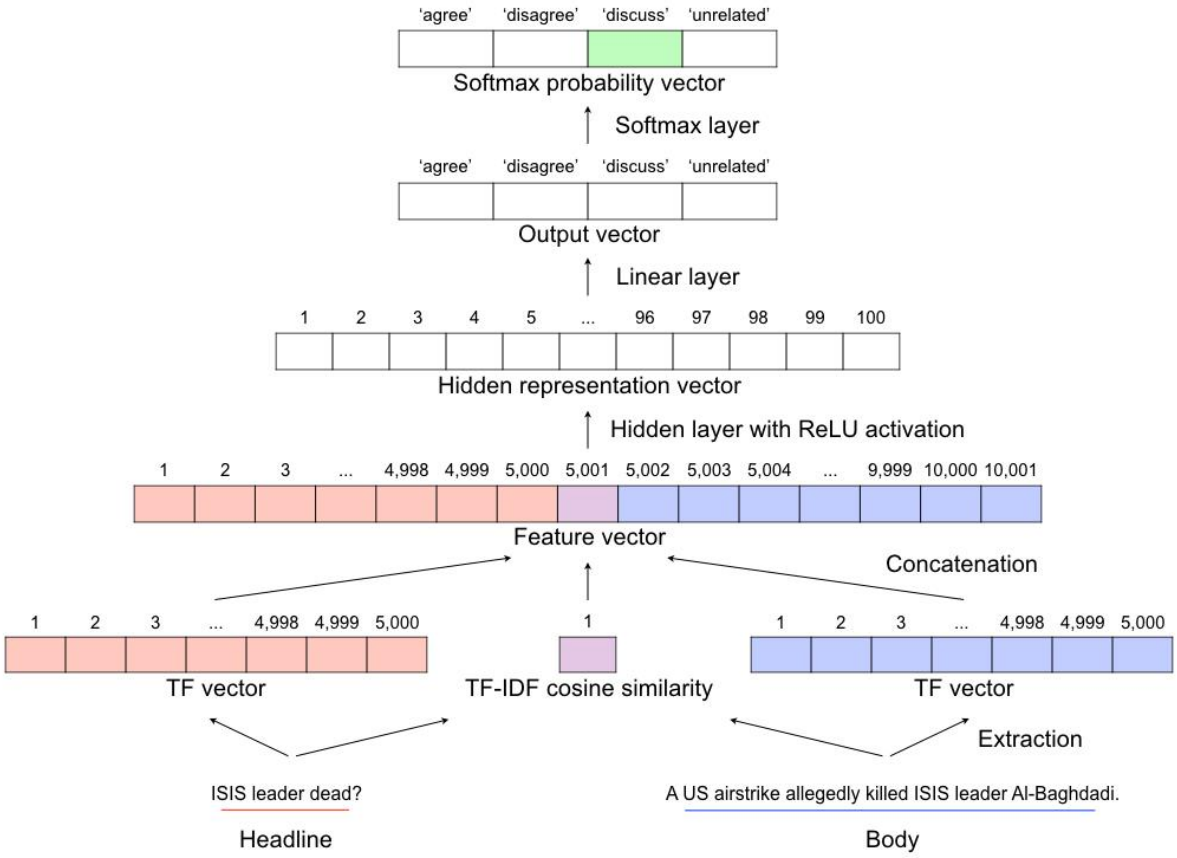


Рис. 1.2. Схематичне зображення механізму роботи алгоритму виявлення позиції у рішенні UCL Machine Reading [15]

Функціональними особливостями даного ПЗ є:

- доволі висока точність виявлення неправдивих новин (81,72 %);
- проста система класифікації, що використовує механізми виявлення фіктивних новин на базі результатів аналізу лексичних ознак;

- детальна інструкція по навчанню моделі і запуску алгоритму класифікації.

Недоліками даного програмного продукту можна вважати:

- відсутність графічного інтерфейсу;
- підтримку лише певних версій програмних бібліотек та мов програмування.

1.3.4. www.factcheck.org

Factcheck.org – веб-сайт, що надає можливість перевірки новини на правдивість. Складається переважно зі статей-спростувань певних фіктивних новин.

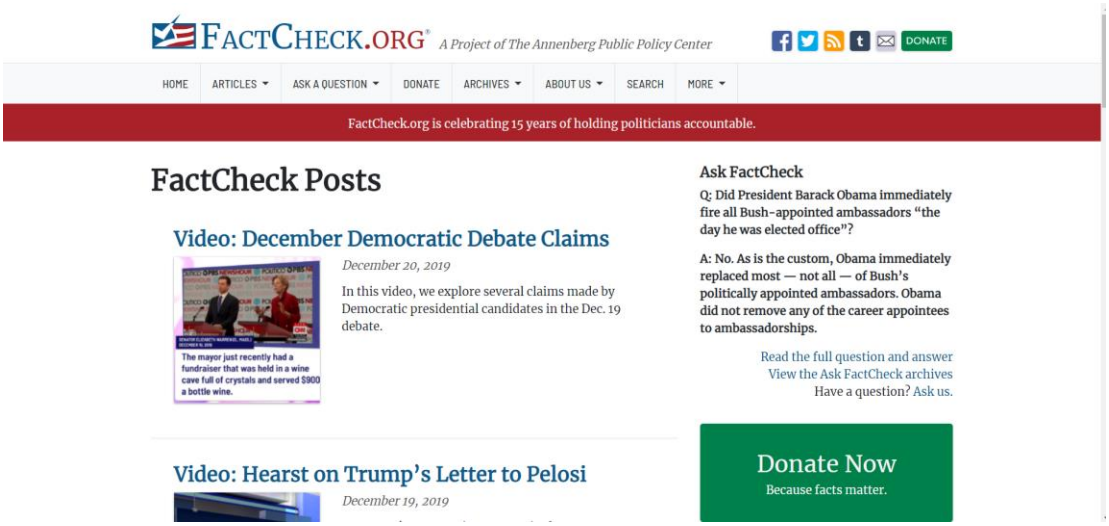


Рис. 1.3. Інтерфейс розділу перевірки новин на правдивість веб-сайту factcheck.org

Функціональними особливостями даного ПЗ є:

- абсолютна точність виявлення фіктивних новин завдяки використанню ручної перевірки кожної окремо взятої новини;
- сфокусованість на джерелах новин Сполучених Штатів Америки;
- результатом класифікації є стаття з детальним аналізом новини;

- зручний інтерфейс користувача.

Недоліками даного програмного продукту можна вважати:

- присвоєння новинам статусу правдивої чи фіктивної відбувається шляхом прямого перегляду новини і пошуками її підтвердження робітниками сайту;
- класифікація новини займає час, за який фіктивні новини і чутки встигають широко поширитися.

1.3.5. Fact Check Explorer

Fact Check Explorer [16] – програмний продукт, запропонований корпорацією Google. Даний програмний продукт створено для фасилітації людей та видань, що перевіряють факти, журналістів та дослідників. Google взагалі не виконує перевірку фактів, а лише виступає у ролі агрегатора вже готових рішень.

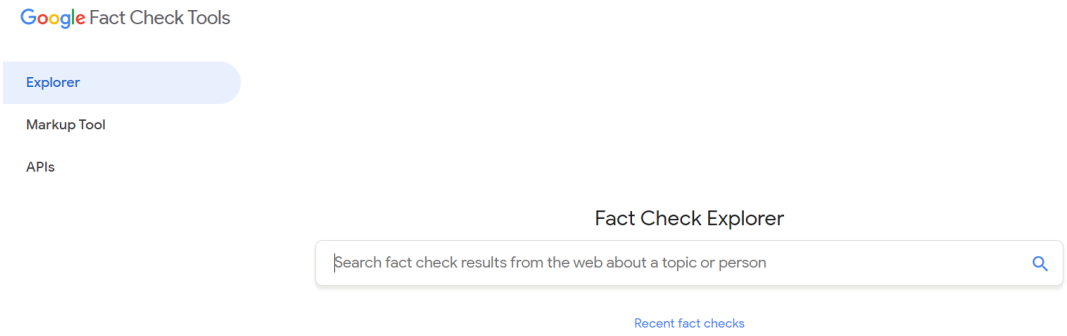


Рис. 1.4. Інтерфейс головної сторінки веб-сайту Fact Check Explorer

Функціональними особливостями даного ПЗ є:

- інтуїтивно зрозумілий програмний інтерфейс;
- доступний пошук новин доступний на більшості мов;
- наявна велика база класифікованих новин.

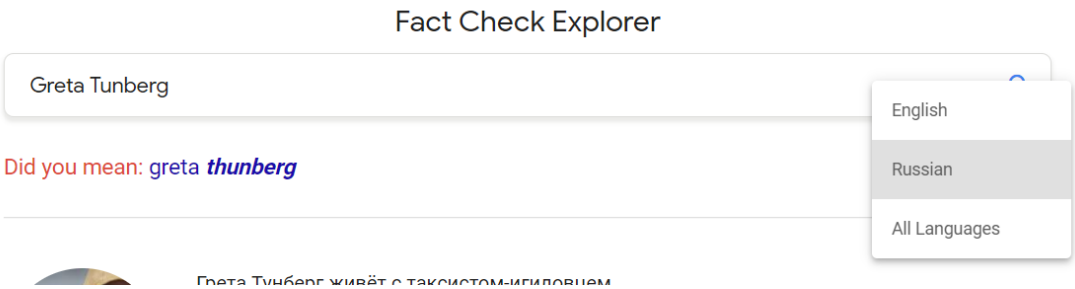


Рис. 1.5. Інтерфейс вибору мови на якій потрібно провести пошук по веб-сайту Fact Check Explorer

Недоліками даного програмного продукту можна вважати:

- повну відсутність механізму класифікації новин;
- залежність термінології, що описує правдивість або фіктивність новини від веб-сайту, з якого походить результат аналізу (відрізняється у кожної новини, рис.1.6);
- залежність правильності виявлення від сайтів-джерел.

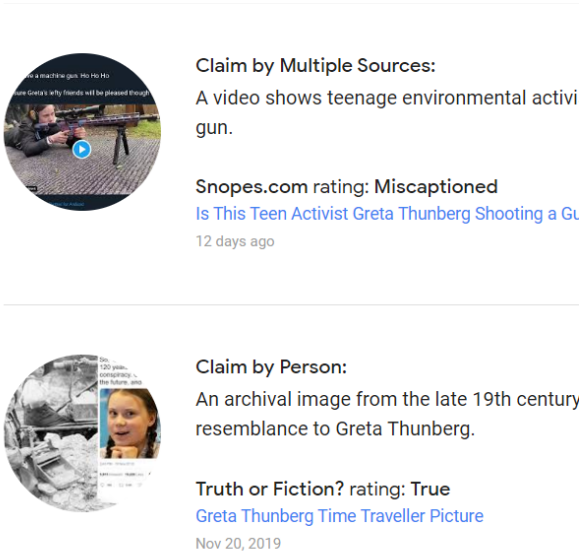


Рис. 1.6. Приклад термінології, що використовують різні сайти-джерела

1.4. Результати проведеного аналізу

Дані, отримані при ретельному аналізі схожих за функціональними можливостями рішень, спрямованих на класифікацію текстових новин, їх

переваг та недоліків, свідчать про те, що жодне з розглянутих рішень в повній мірі не реалізує механізм взаємодії пересічного користувача із програмою.

Проте, кожне з існуючих рішень тією реалізує якусь частину бажаних функціональних особливостей на доволі високому рівні. Наприклад, програмний продукт Talos Intelligence дуже добре реалізує метод комбінованого виявлення фіктивних новин, але через відсутність інтерфейса користувача не дає можливості користування програмою пересічному користувачеві. UCL Machine Reading надає детальну інструкцію користувача, але має великий список вимог до необхідного програмного забезпечення.

З іншого боку веб-сайт www.factcheck.org забезпечує абсолютно точну класифікацію новин, графічний інтерфейс користувача. Проте для класифікації використовується людський ресурс. А в Fact Check Explorer взагалі відсутній механізм класифікації новин, але даний програмний продукт має найзручніший інтерфейс користувача і найбільшу базу класифікованих новин, зібраних в одному місці.

Проаналізувавши наявні аналоги, їх особливості, переваги та недоліки, було сформовано список бажаних функціональних особливостей розроблюваної системи:

1. Система має застосовувати алгоритм машинного навчання з використання лексичного аналізу тексту новини для вирішення проблеми класифікації.
2. Система має надавати передбачення стосовно фіктивності новини.
3. Система має надавати користувачеві простий і зручний інтерфейс.
4. Система має забезпечувати користувачеві доступ до класифікації новин незалежно від характеристик системи і встановлених програмних засобів.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування для розроблення серверної частини

Результатом аналізу функціональних вимог до розроблюваного програмного забезпечення стало рішення розділити реалізацію на два модулі: модуль клієнтської частини і серверний модуль. Серверна частина буде відповідати за оброблення текстової інформації та взаємодію з базою даних. Клієнтська частина буде відповідати за взаємодію користувача з програмним продуктом засобами графічного інтерфейсу.

Мова програмування, що використовується для написання серверної частини, повинна відповідати таким вимогам:

- забезпечувати можливість реалізації функціональної парадигми програмування;
- мати інтерпретовану реалізацію;
- мати фреймворк для написання веб-сервера;
- мати реалізовані бібліотеки для роботи с текстом, його аналізу методами машинного навчання.

2.1.1. JavaScript

JavaScript – динамічна прототипна скриптова мова програмування, найширшого використання набула у браузерах, як мова сценаріїв, що надає інтерактивності веб-сторінкам. JavaScript є найбільш популярною [17] та широко підтримуваною браузерами мовою, що має високий рівень інтеграції з мовою розмітки гіпертексту (HyperText Markup Language) – HTML, та каскадними таблицями стилів (Cascading Style Sheets) – CSS.

JavaScript вважається мультипарадигменною мовою: у ній на однаковому рівні реалізовано об’єкто-орієнтовний, функціональний та імперативний стилі програмування.

Із поширенням JavaScript з’явилася можливість виконувати його код не лише в браузері, але і на сервері, що має спеціальний двигун JavaScript [18].

Двигун, під назвою V8 лежить в основі фреймворку Node.js, який використовується для серверного виконання коду, написаного мовою програмування JavaScript. Фреймворк Node-js містить велику кількість бібліотек та модулів обробки тексту. Крім цього великі бібліотеки, написані іншими мовами програмування, зазвичай мають так звані «обгортки», що дозволяють використовувати їх із js-додатків [19].

Мова JavaScript зазнала впливу багатьох мов: при розробленні стояла мета зробити її схожою на Java, але при цьому легкою для використання людьми без програмістського досвіду [20].

Для JavaScript реалізовано велику кількість бібліотек, зокрема бібліотек для машинного навчання. До таких бібліотек можна віднести ConvNetJS, Brain.js, Deeplearn.js тощо [21].

2.1.2. Java

Java є найбільш використовуваною мовою програмування для написання серверної частини найбільш популярних веб-сайтів [22].

Java – мова програмування загального призначення, що підтримує об’єктно-орієнтовну парадигму програмування, заснована на класах і покликана мати якомога менше залежностей інтерфейса від реалізації.

Однією з головних причин популярності Java є її незалежність від платформи. Це означає, що програми Java можна запускати на різних типах комп'ютерів. Програма Java працює на будь-якому комп'ютері із встановленим середовищем виконання Java, також відомим як JRE. JRE доступний майже для всіх типів комп'ютерів – ПК, на яких працює ОС Windows, комп'ютерів Macintosh, Unix або Linux, величезних мейнфреймів і навіть стільникових телефонів [23].

Java має велику кількість сторонніх бібліотек та API широкого спектру призначення. Для реалізації алгоритмів машинного навчання розробники Java можуть використовувати різні інструменти та бібліотеки. Щонайменше 90 таких проєктів на базі Java перераховані лише на MLOSS.org [24].

Однак Java є надмірним засобом реалізації для багатьох невеликих додатків, де можна досягти такого ж результату, використовуючи простіші рішення. Однією з сильних сторін мови є потужність з точки зору масштабованості.

2.1.3. C#

C# – це мультипарадигменна [25] мова загального призначення для розробки програм на платформі Microsoft. Необхідною умовою роботи мови є встановлений у системі фреймворк .NET.

C# можна використовувати для створення програмного забезпечення майже будь-якого спрямування, але найчастіше його використовують для створення настільних програм та ігор для ОС Windows. Мова C# також може бути використана для розробки веб-додатків і наразі стає все більш популярною для виконання розробок під мобільні платформи. Інструменти, подібні до фреймворку Xamarin дозволяють додаткам, написаним на C#, виконуватися майже на будь-якому мобільному пристрої [26].

Розширення платформи .NET з інструментами та бібліотеками для створення веб-додатків називається ASP.NET.

ASP.NET розширює властивості платформи .NET і додає до неї наступні можливості [27]:

- синтаксис шаблонів веб-сторінок під назвою Razor, що забезпечує створення динамічних сторінок за допомогою C#;
- базовий фреймворк для обробки веб-запитів в C# або F#;

- система аутентифікації, що включає базу даних, бібліотеки та сторінки шаблонів для обробки входів, включаючи багатофакторну аутентифікацію та зовнішню аутентифікацію за допомогою Google, Twitter тощо;
- бібліотеки загальних веб-шаблонів, таких як контролер перегляду моделі (MVC) ;
- розширення для редактора, що забезпечують виділення синтаксису, автодоповнення коду та інші функціональні можливості для розроблення веб-сторінок.

Остання версія фреймворку ASP.NET Core отримала можливість запуску на операційних системах macOS та Linux [28].

Для роботи із машинним навчанням може бути використано одну із безлічі спеціалізованих бібліотек [29]. Також API для C# має одна з найпопулярніших бібліотек машинного навчання, розроблена компанією Google – TensorFlow [30, 31].

2.1.4. Ruby

Ruby – інтерпретована динамічна мова програмування високого рівня, загального призначення і відкритим сирцевим кодом та акцентом на простоту та продуктивність.

Ruby використовує механізм збирання сміття (garbage collection), підтримує декілька парадигм програмування, включаючи функціональне, об'єктно-орієнтоване та процедурне програмування.

Оскільки Рубі – динамічна мова, одна і та сама інформація може легко означати різні речі, в залежності від контексту. Із ростом додатку, написаного на Ruby, його стає складніше підтримувати, оскільки помилки важко відстежувати і виправляти. Також за рахунок динамічності і гнучкості, Ruby є доволі повільною мовою.

Ruby є досить популярною мовою. Так, спільнота Ruby на StackOverflow налічує більш ніж 27 тисяч учасників і є дев’ятою найбільшою на ресурсі [32].

Мова налічує велику кількість зовнішніх модулів і фреймворків для розробки серверної складової проєкту.

Переглядаючи бібліотеки для машинного навчання, можемо зробити висновок, що вибір не такий багатий, а наявні рішення недостатньо задокументовані.

Вони також не забезпечують достатньо високої швидкості обчислень. Ці фактори приводять до висновку, що у використанні Ruby для машинного навчання більше ризиків, ніж переваг.

2.1.5. PHP

PHP – це динамічна та інтерактивна серверна об'єктно-орієнтована, скриптова мова з відкритим кодом. Як і будь-яка інша скриптова мова, вона дозволяє розробникам вбудовувати логіку у створення вмісту веб-сторінок та обробляти дані, повернені веб-браузером. PHP також містить ряд розширень, які полегшують взаємодію з БД, дістаючи дані для відображення на веб-сторінці та зберігаючи інформацію, введену відвідувачем веб-сайту, назад у базу даних.

PHP інтегрований з низкою популярних БД, включаючи PostgreSQL, MySQL, Oracle, Informix та Microsoft SQL Server.

Як і інші скриптові мови, PHP дозволяє веб-розробникам визначати поведінку та логіку, необхідну їм на веб-сторінці. Ці сценарії вбудовані в документи HTML, які обслуговуються веб-сервером. Інтерпретатор має форму модуля, який інтегрується у веб-сервер, перетворюючи скрипти в команди, які комп'ютер потім виконує для досягнення результатів, визначених у скрипті веб-розробником.

У вільному доступі є велика кількість бібліотек і модулів різного спрямування. Серед веб-майстрів найбільшою популярністю користуються такі фреймворки, як Symphony, Zend Framework, Yii тощо.

2.1.6. Python

Python – інтерпретована високорівнева мова програмування загального призначення. Python динамічна мова, що використовує механізми збірки сміття (garbage collector) і підтримує декілька парадигм програмування, включаючи функціональне, структуроване та об'єктно-орієнтоване програмування.

Можна виділити основні характеристики, що сприяли популярності мови:

- легкість написання коду, завдяки зрозумілому синтаксису і динамічній типізації. Читати код легко по тим самим причинам;
- кросплатформність досягається доступністю інтерпретатора мови на більшості сучасних операційних систем;
- механізм роботи інтерпретатора мови полегшує процес відлагодження програми;
- наявність великої кількості відкритих бібліотек (бібліотеки для регулярних виразів, створення документації, тестування, реалізації потоків, баз даних, електронної пошти, маніпулювання зображеннями, створення графічного інтерфейсу, бібліотеки з математичними функціями, та реалізаціє алгоритмів машинного навчання тощо)
- інтегровність мови надає можливість використовувати код, написаний мовою Python із програмних середовищ, написаних на C, C++, Java і навпаки.

Для написання серверної частини додатку мовою Python існує безліч рішень. Найпопулярнішим із них є Flask – фреймворк для створення веб-

додатків. Інший популярний фреймворк – Django містить більше вбудованих модулів, має ширший функціонал і дещо вищу складність.

Для даної мови програмування реалізованою є одна з найкращих бібліотек для машинного навчання – TensorFlow.

Таблиця 2.1

Порівняння мов програмування для розроблення серверної частини

Мова	Інтерпретована модель виконання	Функціональна парадигма	Бібліотеки для роботи з текстом методами ML	Наявність фреймворку для реалізації веб-серверу
JavaScript	+	+	+	+
Java	–	–	+	+
C#	–	–	+	+
Ruby	+	+/–	+/–	+
PHP	+	–	+/–	+
Python	+	+/–	+	+

Після проведення аналізу найбільш поширених мов програмування для реалізації серверної частини було обрано мову Python, яка не тільки відповідає всім встановленим вимогам, а й має переваги у простоті синтаксису та наявності бібліотек високої якості з функціоналом, необхідним для машинного навчання та обробки тексту.

2.2. Вибір технології для розроблення клієнтської частини

Основними вимогами до технології, що використовується для розроблення клієнтської частини є:

- швидкість завантаження;
- доступність кінцевому користувачеві;

- легкість в налаштуванні.

2.2.1. Веб-сайт

Веб-сайт – це сукупність веб-сторінок і їхнього вмісту, що ідентифікується загальним доменним іменем і публікується щонайменше на одному веб-сервері.

Усі загальнодоступні веб-сайти колективно складають Всесвітню мережу. До деяких веб-сайтів доступ можна отримати лише в приватній мережі, такі веб-сайти називаються приватними. Прикладом може слугувати внутрішній веб-сайт для працівників компанії.

Основні переваги використання веб-сайту у якості клієнтської частини роботи:

- користувачі можуть отримати доступ до веб-сайтів на різноманітних пристроях, включачи смартфони, планшети, ноутбуки та настільні комп'ютери – можливість отримати доступ обмежується лише наявністю веб-браузера на пристрої;
- наявна велика кількість фреймворків для забезпечення динамічності сторінок. Найпопулярнішими є React, AngularJS та Vue.js.

Недоліки використання веб-сайту для взаємодії з кінцевим користувачем:

- необхідність переходу на окрему веб-сторінку для того, щоб перевірити новину на правдивість.

2.2.2. Чатбот для месседжера

Чатбот – це програмне забезпечення, яке використовується для ведення онлайн розмови в чаті за допомогою тексту чи текстового мовлення, замість забезпечення прямого контакту з живим людським агентом [33]. Створені для переконливого моделювання способу поведінки

людини як розмовного партнера, чат-системи, як правило, вимагають постійного налаштування та тестування, а багато виробників залишаються нездатними адекватно переробити або пройти галузевий стандарт тестування Тьюрінга.

Чатботи зазвичай використовуються в діалогових системах для різних цілей, включаючи обслуговування клієнтів, маршрутизацію запитів або для збору інформації. У той час як деякі додатки для роботи в чаті використовують обширні процеси класифікації слів, обробники мови та складний AI, інші просто сканують на наявність загальних ключових слів та генерують відповіді, використовуючи загальні фрази, отримані з пов’язаної бібліотеки чи бази даних.

Позитивні аспекти використання чатботів для клієнтської частини додатку:

- велика аудиторія користувачів месенджерів. Одним із найпопулярніших месенджерів є Facebook Messenger, він налічує 1,2 мільярди користувачів, тоді як аудиторія, більш популярного в Україні, месенджера Telegram нараховує 400 мільйонів користувачів;
- наявність відкритих API для створення ботів із повною документацією процесу розроблення;
- простий механізм комунікації бот-користувач.

Негативні аспекти використання чатботу як клієнтської сторони додатку:

- необхідність виходу із сайту-джерела новини для того, щоб протестувати скопійований текст у месенджері.

2.2.3. Розширення до браузера

Розширення – це невеликий програмний модуль для збільшення функціональних можливостей веб-браузера. Браузери, як правило,

дозволяють використовувати різні розширення, включаючи зміни інтерфейсу користувача, блокування реклами та керування файлами cookie.

Існує також поняття плагіну для браузера. Основна відмінність плагіна від розширення полягає в тому, що розширення зазвичай є лише вихідним кодом, але плагіни – це завжди виконувані файли (тобто об'єктний код). Станом на 2020 рік, у більшості браузерів плагіни вважаться застарілими, тоді як розширення широко використовуються. У найпопулярнішому веб-переглядачі Google Chrome доступні тисячі розширень, але лише один плагін: Adobe Flash Player (який за замовчанням вимкнено).

Переваги використання розширень до браузера у якості клієнтської сторони додатку:

- легкість у написанні, наявність розгорнутої документації;
- доступність кінцевому користувачеві;
- можливість використання не відриваючись від процесу читання новин.

Недоліки використання розширення як програмного рішення:

- можливість використання на мобільних пристроях за замовчанням відсутня, проте використання розширень для браузеру Chrome можливе у таких мобільних браузерах, як Kiwi або Yandex Browser.

Оцінивши позитивні та негативні аспекти використання усіх запропонованих технологій, було вирішено реалізувати клієнтську частину додатку у вигляді розширення для браузера. Оскільки найпопулярнішим браузером на даний момент є Google Chrome [34], саме цей браузер було вибрано для написання розширення.

2.3. Вибір СКБД

СКБД розроблюваної системи повинна забезпечувати збереження текстових новин із мітками про правдивість або фіктивність. Виходячи із характеру даних, збереження яких необхідно забезпечити, було виділено наступні вимоги до СКБД:

- безкоштовність;
- наявність індексів;
- підтримка повнотекстового пошуку;
- гнучкість у формуванні запитів;
- використання нереляційного підходу.

2.3.1. Redis

Redis – це розподілене сховище типу «ключ-значення», що зберігаються в оперативній пам’яті. Забезпечує реалізацію різних видів абстрактних структур даних: рядків, списків, асоціативних масивів, наборів, відсортованих наборів, бітових асоціативних масивів, потоків та просторових індексів.

У сховищі типу «ключ-значення» використовується хеш-таблиця, в якій унікальний ключ вказує на певний елемент. Ключі можна організувати в логічні групи, за умови унікальності ключів у власній групі. Це дозволяє існуванню однакових ключів у різних логічних групах.

Основні недоліки Redis:

- розмір БД обмежений доступною пам'яттю;
- шардінг веде до збільшення затримки;
- LUA скриптинг використовується, як заміна SQL;
- відсутність сегментації на осіб чи груп. Відсутній контроль доступу.

Основні переваги Redis:

- швидкість: зберігання даних в пам'яті робить швидше роботу з ними;
- скрипти LUA виконуються безпосередньо в пам'яті, що також прискорює роботу;
- зручні формати запитів в даних (Geospatial, Hyperloglog, Hash тощо);
- алгоритми старіння даних;
- наявність можливості розгортання у хмарному середовищі.

Redis рекомендується до використання у випадку, коли необхідним є вкрай швидкий доступ до даних. Особливість збереження даних у кеші дозволяє швидко здійснювати операції читання та запису. Redis доречно використовувати в якості оболонки для обробки часто запитуваних даних, яка позбавляє від необхідності здійснення частих запитів до самої бази.

2.3.2. Apache Cassandra

Cassanra – це безкоштовна відкрита розподілена нереляційна СКБД, призначена для обробки великого обсягу даних, кластеризованих по багатьох датацентрах, що забезпечує високу доступність без точкових відмов. Cassandra підтримує асинхронну нецентралізовану реплікацію, що забезпечує користувачеві доволі високу швидкість доступу до даних.

У Cassanra дані зберігаються у стовпцях, на відміну від зберігання у рядках у більшості реляційних СКБД.

Переваги:

- масштабованість: пропускна здатність для читання та запису збільшується лінійно, оскільки нові машини додаються не перериваючи роботу додатка;

- відмовостійкість: дані автоматично реплікуються у кілька вузлів. Підтримується реплікація в декількох датацентрах. Пошкоджені вузли можна замінити без простоїв;
- висока швидкість: рядки, що відповідають одному стовпцю, зберігаються як окремий запис на диску, що сприяє високій швидкості доступу під час операцій читання та запису;
- простий інтерфейс: для доступу до бази даних використовується альтернативна до SQL – мова запитів до Cassandra (CQL).

СКБД Cassandra використовується для зберігання послідовних даних, логів, або великого обсягу інформації, який може генеруватися. Дану СКБД доречно використовувати для запису великих масивів даних, за умови, що звернень для читання буде не багато, а самі дані будуть не пов'язані ід собою.

2.3.3. MongoDB

MongoDB – кросплатформна документо-орієнтована NoSQL СКБД, що надає такі можливості, як: забезпечення гнучкої моделі даних, індексація та швидкісний запит, що робить навчання та використання алгоритмів машинного навчання набагато простішим, ніж з традиційними реляційними базами даних. Запуск MongoDB як резервної бази даних для зберігання наборів даних для машинного навчання сприяє підвищенню ефективності.

Переваги використання MongoDB:

- легка у масштабуванні;
- є безкоштовною;
- для зберігання даних, що використовуються в даний момент, застосовується внутрішня пам'ять, що забезпечує більш швидкий доступ;
- підтримує динамічні запити документів (document-based query);
- збереження даних у вигляді JSON документів;

- відсутня необхідність маппінгу об'єктів додатку в об'єкти БД;
- додавання та оновлення документів відбувається досить швидко;
- проста установка, реплікація (replica sets) та масштабування (sharding).

Недоліки використання MongoDB:

- високий обсяг використовуваної пам’яті: MongoDB зберігає назви ключів для кожної пари значень. Крім того, через відсутність функціональності операцій JOIN, може формуватися надмірність даних, що і призводить до збільшення обсягу використання пам'яті;
- обмежений розмір даних: розмір документа не повинен перевищувати 16 мегабайт;
- обмежене вкладення: не можна зробити вкладення документів на більш ніж 100 рівнів.

MongoDB використовується для зберігання JSON-документів в колекціях і здійснення запитів по необхідних полях. Дану СКБД доречно використовувати для створення додатків, в даних яких не буде міститися занадто великої кількості зв’язків.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ СИСТЕМИ

3.1. Опис вимог до розроблюваної системи

Процес збору вимог є першим кроком в процесі створення програмного забезпечення. Основним завданням етапу збору вимог є фіксування загального образу продукту. Образ продукту описує, що повинен являти собою продукт. Границі проєкту показують, до якої області довгострокового образу продукту спрямований поточний проєкт. Образ продукту має на меті визначення довгострокових перспектив розвитку проєкту, в той час як границі проєкту можуть бути визначені для конкретної ітерації. Слід дуже уважно ставитися до формування образу і границь проєкту. Це дозволить уникнути або, принаймні, запланувати на більш пізній час реалізацію менш критичних для успіху проєкту можливостей програмного забезпечення.

Зібрані вимоги оформляються у вигляді специфікації вимог (System Requirement Specification – SRS). Основними помилками, що допускаються етапі первинного збору вимог, є [35]:

- нечітке визначення границь проєкту, що призводить до затягування термінів виконання та збільшення об’єму витрат за рахунок включення в продукт другорядних можливостей;
- недостатньо чітко визначені групи користувачів продукту;
- затягування роботи над проєктом шляхом спроби досконального опису усіх вимог;
- відсутність пріоритизації вимог, що робить реалізацію в першу чергу найнеобхідніших вимог, відкладаючи реалізацію другорядного функціоналу на більш пізній термін.

Отже, розглянемо, які саме типи вимог існують:

- функціональні вимоги: детально висвітлюють функціональність продукту;

- нефункціональні вимоги: не пов'язані з "функціональністю" продукту, але охоплюють такі характеристики, як надійність, масштабованість, безпека, інтеграція тощо;
- бізнес вимоги: описують характеристики запропонованої системи з точки зору кінцевого споживача системи.

Специфікації вимог повинні включати такі розділи:

- призначення: визначення, огляд системи та передумови;
- загальний опис: Припущення, обмеження, бачення продукту;
- конкретні вимоги. Атрибути системи, функціональні вимоги, вимоги до бази даних.

Важливо зробити специфікації читабельними для всіх зацікавлених сторін.

Сценарії використання описують взаємодію користувача із системою, що необхідна для досягнення конкретних цілей.

Кожен сценарій виконання включає три основні елементи:

- дієва особа: користувач, що взаємодіє із системою;
- система: описується функціональними вимогами, що визначають передбачувану поведінку продукту;
- цілі: мета взаємодії користувача з системою.

Для представлення прецедентів використання є два формати:

- сценарій використання, записаний у текстовому вигляді;
- діаграма прецеденту.

Специфікація сценарію використання представляє послідовність подій разом з суміжною інформацією, що стосується цього шляху використання ПЗ.

Діаграма сценаріїв використання містить обмежену кількість деталей і є простою для сприйняття. Вона показує огляд взаємовідносин між суб'єктами, різними випадками використання та системою.

Діаграма сценаріїв використання включає такі основні елементи:

- сценарій використання: описує конкретну дію, що користувач може виконати у системі;
- межі системи: зазвичай представлені прямокутником, в якому групуються різні випадки використання системи;
- суб’єкт: зовнішні користувачі системи. Це можуть бути як люди, так і інші системи;
- зв’язки: зображуються лініями, що показують різні типи стосунків між суб’єктами та сценаріями використання.

Призначення та загальний опис системи міститься у розділах 1 і 2 даної дипломної роботи. Наразі виділимо функціональні вимоги, що є першочерговими для реалізації:

1. Система дозволяє користувачу проаналізувати довільний об’єм тексту на правдивість.
2. Система має функцію редагування текстової новини, вставленої в поле перевірки.
3. Система надає результат перевірки новини у вигляді прапорів «Real» або «Fake» у випадку, відповідно, правдивості або фіктивності новини.
4. Система надає рейтинг сайту, з якого взято новину, з точки зору правдивості матеріалів відповідно до рейтингових таблиць спеціалізованих сайтів.
5. Можливим є доступ до основного функціоналу системи за допомогою API.

Проілюструємо сформульовані вимоги відповідними діаграмами сценаріїв використання.

Сценарій використання 1: «Виконання аналізу тексту» – описує процес перевірки текстової новини на фіктивність (рис. 3.1).

1. Користувач виділяє фрагмент або цілу новину, яку потрібно перевірити. (Опціональний пункт, якщо проаналізувати необхідно текст, що виділено на сторінці)
2. Користувач активує додаток.
3. Користувач робить поле вводу активним.
4. Користувач вставляє текст, який необхідно проаналізувати. Або (за умови виконання кроку 0) користувач натискає на кнопку «Скопіювати виділення» і бачить, що виділений текст з’явився в полі вводу.
5. Користувач натискає на кнопку «Проаналізувати» і отримує результат.

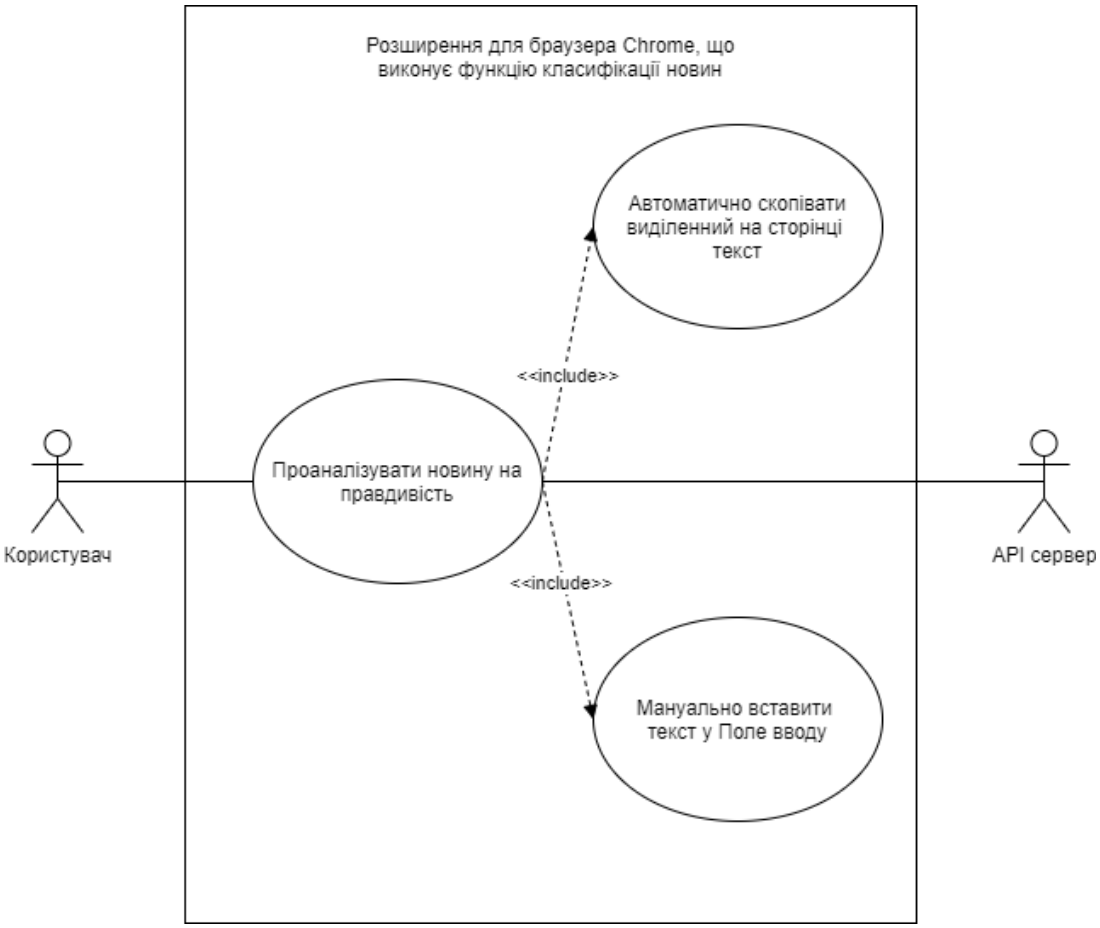


Рис. 3.1. Сценарій використання «Виконання аналізу довільного тексту»

Сценарій використання 2: «Редагування тексту новини перед аналізом» – описує процес редагування користувачем тексту новини перед виконанням аналізу (рис. 3.2).

1. Користувач активує додаток.
2. Користувач робить поле вводу активним.
3. Користувач вставляє текст, який необхідно проаналізувати, самостійно, або користуючись кнопкою «Вставити виділення».
4. Користувач редагує зміст поля вводу.

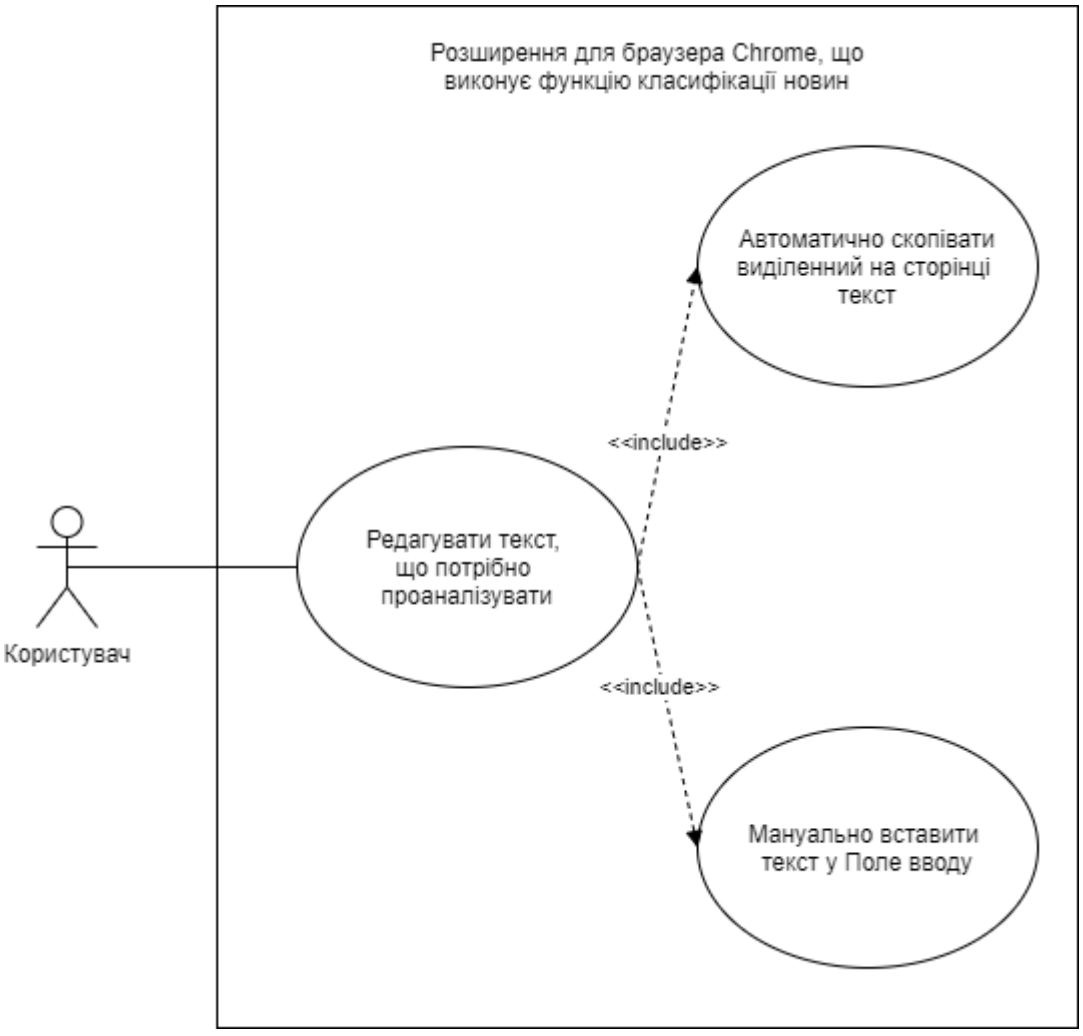


Рис. 3.2. Сценарій використання «Редагування тексту новини перед аналізом»

Наступним кроком сформулюємо нефункціональні вимоги до розроблюваної системи, результат оформимо у вигляді таблиці:

Таблиця 3.1

Нефункціональні вимоги

Код	Опис вимоги
NF-1	Текст новини, що надається користувачем для аналізу необхідно векторизувати. До аналізу повинно бути надано векторизований словник максимальною довжиною 120 слів.
NF-2	Точність класифікації новин як правдивих або фіктивних, повинна складати не менш, ніж 80%.
NF-3	Максимально можлива затримка при виконанні запиту на аналіз – 10 секунд.
NF-4	Система повинна бути пристосована до навантаження у 800 одночасних запитів.
NF-5	Розширення для веб-браузера Google Chrome повинно коректно працювати на таких платформах, як Windows, Linux, macOS та ChromeOS.
NF-6	Серверна частинна має бути розміщена на хмарній платформі, що знаходиться під управлінням Linux Ubuntu.
NF-7	База даних (MongoDB) повинна бути розгорнута у хмарній службі баз даних.

3.2. Опис структурної організації системи

Кінцева розроблена система складається з п’яти модулів (рис. 3.3), розглянемо призначення кожного з цих модулів.

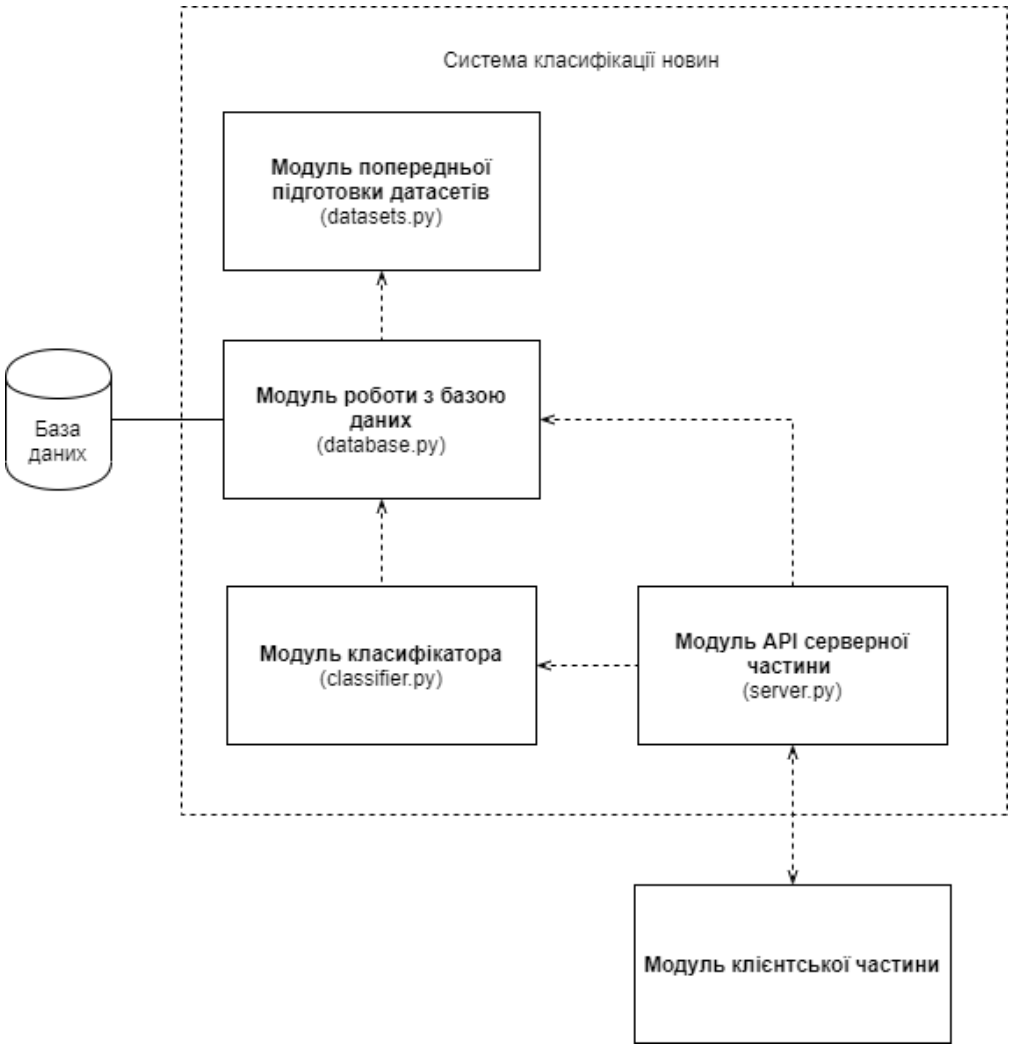


Рис. 3.3. Загальна структура системи

3.2.1. Модуль попередньої підготовки набору даних

В роботі використано два набори даних. Модуль використовується для формування композитного набору даних із двох наборів даних (FakeNewsNet та Fake and real news dataset), а це значить, що у кінцевій системі він не використовується, на відміну від результату його роботи.

3.2.2. Модуль класифікатора

Даний модуль відповідає за побудову, навчання і тестування моделі, а також за збереження словника, та його векторного представлення, отриманого в результаті навчання моделі, збереження самої моделі та її оптимізатора виконання класифікації конкретної новини.

3.2.3. Модуль клієнтської частини

Даний модуль відповідає за графічний інтерфейс взаємодії користувача із системою. Клієнтську частину було вирішено реалізувати у вигляді односторінкового додатку (англ. Single Page Application). Односторінковий додаток – це веб-додаток, який взаємодіє з веб-браузером, динамічно переписуючи поточну веб-сторінку новими даними з веб-сервера, замість методу браузера за замовчуванням завантаження цілих нових сторінок. Мета даної особливості реалізації в тому, щоб пришвидшити оновлення контенту сторінки. В односторінковому додатку всі необхідні HTML, JavaScript та CSS-код браузер отримує після завантаження однієї сторінки, або ж відповідні ресурси динамічно завантажуються та додаються на сторінку за необхідності, як правило, у відповідь на дії користувача. Сторінка не завантажується в будь-який момент процесу, а також не передає керування на іншу сторінку, хоча хеш місцеположення або API історії HTML5 можна використовувати для забезпечення сприйняття та навігації окремих логічних сторінок у програмі.

3.2.4. Модуль роботи з базою даних

У даному модулі описані функції роботи із базою даних системи, операції додавання документів; реалізовано функції для запиту інформації з БД як у загальному вигляді, так і за такими критеріями як Джерело набору даних та Призначення набору даних.

3.2.5. Модуль API серверної частини

Відповідає за виконання зв’язуючої функції для модулів клієнтскої частини, класифікатора та модуля роботи з базою даних. При реалізації API було вирішено використовувати REST-підхід. REST (від англ. Representational State Transfer – «передача репрезентативного стану») – архітектурний стиль взаємодії компонентів розподіленого додатка в мережі.

REST являє собою узгоджений набір обмежень, що враховуються при проєктуванні розподіленої гіпермедіа-системи. У певних випадках це призводить до підвищення продуктивності і спрощення архітектури. У широкому сенсі компоненти в REST взаємодіють на зразок взаємодії клієнтів і серверів у Всесвітній павутині. REST API повинно задовільняти наступні критерії:

- використання клієнт-серверної архітектури;
- відсутність збереження стану клієнта на сервері між запитами;
- клієнти можуть виконувати кешування відповіді сервера;
- забезпечення одноманітності інтерфейсу;
- використання проміжних серверів для підвищення масштабовності;
- код по запиту. Надає можливість клієнту завантажувати програмний код з сервера у вигляді сценаріїв або аплетів.

3.3. Опис структур даних системи

В результаті аналізу характеру даних, збереження яких необхідно забезпечити, було виділено такі основні сутності бази даних:

Веб-сайт (Website):

- домен (domain) – доменне ім’я сайту;
- тип (type) – тип сайту, відповідно до класифікації новин, які на ньому публікують;
- джерело (source) – джерело прапору класифікації.

Новина (Article):

- текст (text) – текстова складова новини;
- лейбл (label) – маркування новини. Приймає значення 1, якщо новина правдива, у іншому випадку 0;
- назва набору даних (dataset_name).

Слово (Word):

- векторне представлення (vec) – вага слова, отримана в результаті навчання класифікатора;
- безпосередньо слово (word);
- назва тренувального набору даних (train_dataset_name) – назва набору даних, на новинах якого було натреновано модель;
- алгоритм тренування (train_algorithm) – алгоритм, який використовувався для тренування класифікатора.

3.4. Алгоритм класифікації новин

У даній роботі будемо використовувати методи контрольованого машинного навчання, яке також називають навчання з учителем (англ. Supervised learning). За умови контрольованого навчання існує множина вхідних змінних X, та вихідна змінна Y, використовується певний алгоритм для знаходження функції відображення Y = f(X). Мета методу полягає в тому, щоб визначити функцію відображення настільки добре, щоб можливим стало передбачення вихідних змінних Y для довільних вхідних даних X.

Методи алгоритмів керованого машинного навчання включають лінійну та логістичну регресію, багатокласну класифікацію, дерева рішень та вектор опорних векторів. Для тренування алгоритму методами контрольованого навчання необхідною є наявність розміченого набору даних, в якому будуть міститися правильні відповіді, поряд із вхідними даними. Наприклад, алгоритм класифікації емоцій на фотографіях зможе ідентифікувати конкретні емоції після того, як його навчать на наборі промаркованих мітками емоцій зображень, що також містять певні ідентифікаційні характеристики.

Методами контрольованого навчання можна вирішувати дві великі групи проблем: проблеми регресії та класифікації. Обидві проблеми мають

за мету побудову лаконічної моделі, яка може передбачити значення залежного атрибута зі змінних атрибутів. Різниця між двома завданнями полягає в тому, що залежний атрибут є числовим для регресії та категоричним для класифікації.

Проблема, що підіймається у даній роботі, відноситься до множини проблем класифікації.

Проблеми класифікації характеризуються тим, що вихідним значенням є конкретна категорія із переліку можливих категорій. Класифікаційна модель намагається передбачити значення одного або декількох результатів із вхідних значень.

Класифікація або прогнозує категоричні мітки класів, або класифікує дані (будує модель) на основі навчального набору та значень (мітки класів) у класифікаційних атрибутах та використовує їх у класифікації нових даних. Існує ряд моделей класифікації. Класифікаційні моделі включають логістичну регресію, дерево рішень, дерево з підсиленим градієнтом, багатошаровий персептрон, наївний баєсів класифікатор тощо.

У даній роботі навчальним набором слугують датасет новин, а мітками класів – позначки про фіктивність або правдивість новини (0 та 1, відповідно, у полях label для кожної новини набору даних). Для вирішення проблеми класифікації, будемо використовувати модель багатошарового персептрону.

Схема алгоритму навчання класифікатора на високому рівні абстракції приведена на рис. 3.4.

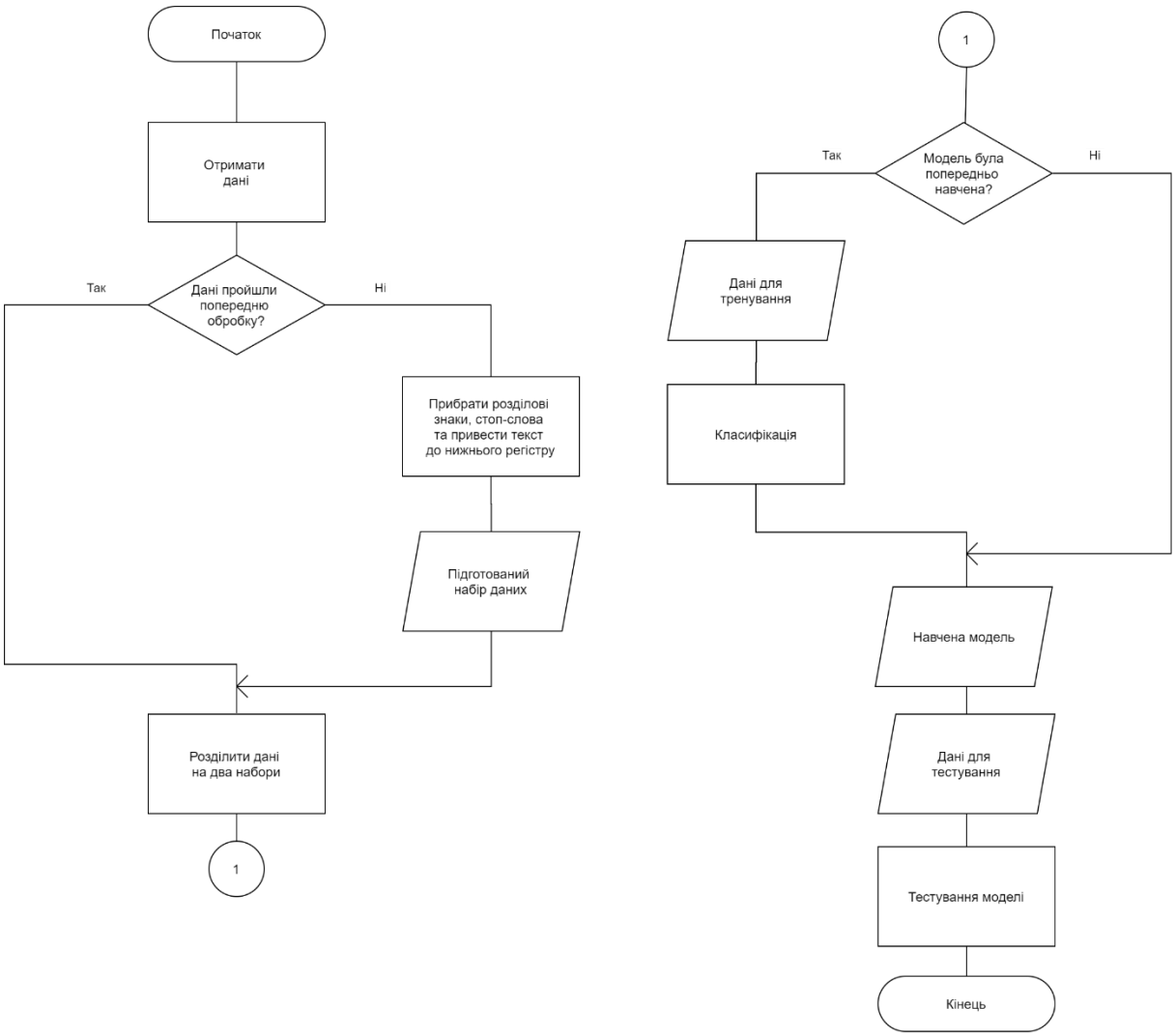


Рис. 3.4. Навчання класифікатора. Схема алгоритму

4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Реалізація клієнт-серверної взаємодії

Серверну частину програмного забезпечення реалізовано у вигляді серверного додатку, розгорнутого на хостингу Heroku. Веб-додаток реалізовано мовою Python із використанням фреймворку Flask та Gunicorn WSGI (англ. Web Server Gateway Interface) HTTP сервера. Даний додаток реалізує архітектурний стиль побудови програмного забезпечення – REST (англ. Representational state transfer), що забезпечує набір обмежень, відповідно до якого буде відбуватися клієнт-серверна взаємодія. Архітектурні обмеження, що накладає дана модель взаємодії, наступні:

- жорстке розділення користувацького інтерфейсу і сховища даних, що підвищує переносимість і гнучкість системи, покращує масштабованість;
- відсутність збереження стану системи: зв’язок клієнт-сервер обмежується тим, що на сервері між запитами не зберігається контекст клієнта. Кожен запит будь-якого клієнта містить всю інформацію, необхідну для обслуговування запиту, і стан сеансу зберігається у клієнта;
- використання кеша, що сприяє підвищенню продуктивності системи шляхом скорочення запитів до сервера;
- використання проксі-серверів, що також напрямлене на те, щоб покращити масштабованість системи.

Даний REST API забезпечує дві основні функції системи:

- надання рейтингу сайту відповідно до правдивості матеріалів, що публікуються на ресурсі;
- аналіз текстової новини на правдивість.

Розглянемо детальніше шляхи реалізації наведених функціональних можливостей.

4.2. Збір та аналіз даних

В роботі використано композитний набір даних, що сформовано з двох наборів, розглянемо їхню структуру.

Перший набір даних (Fake and real news dataset [36]) оформлено у вигляді двох файлів: True.csv та Fake.csv, що містять, відповідно, правдиві та фіктивні новини. До даного набору даних входять переважно політичні новини.

Другий набір даних (FakeNewsNet [37]) має дещо складнішу організаційну структуру (рис. 4.1), новини в ньому зберігаються в окремих папках, а папки сортовано одночасно за правдивістю та джерелом новини. До набору даних FakeNewsNet входять новини, вручну класифіковані робітниками веб-сайтів politifact.com та gossipcop.com.

Рис. 4.1. Приклад набору даних FakeNewsNet

За збірку обох наборів даних та приведення їх до однакового формату відповідає модуль dataset_utils.py. Основна функція, get_all_news(), повертає чотири списки: по списку текстів новин та їхніх міток (правдивою чи фіктивною є дана новина) для тренування та тестування.

Композитний набір даних містить 65746 розмічених текстових новин, серед яких 37115 є правдивими, а фіктивними, відповідно, є 28631 новини. Даний набір розділено навпіл задля формування наборів для тренування та тестування моделі класифікації (рис. 4.2). За складом правдивих і текстових новин та розподілом по джерелам, набір даних для тренування є ідентичним набору даних для тестування тренованої моделі. Так, у тренувальному наборі даних міститься 18556 правдивих та 14314 фіктивних новин, а в тестовому, відповідно, 18559 та 14317 новин.

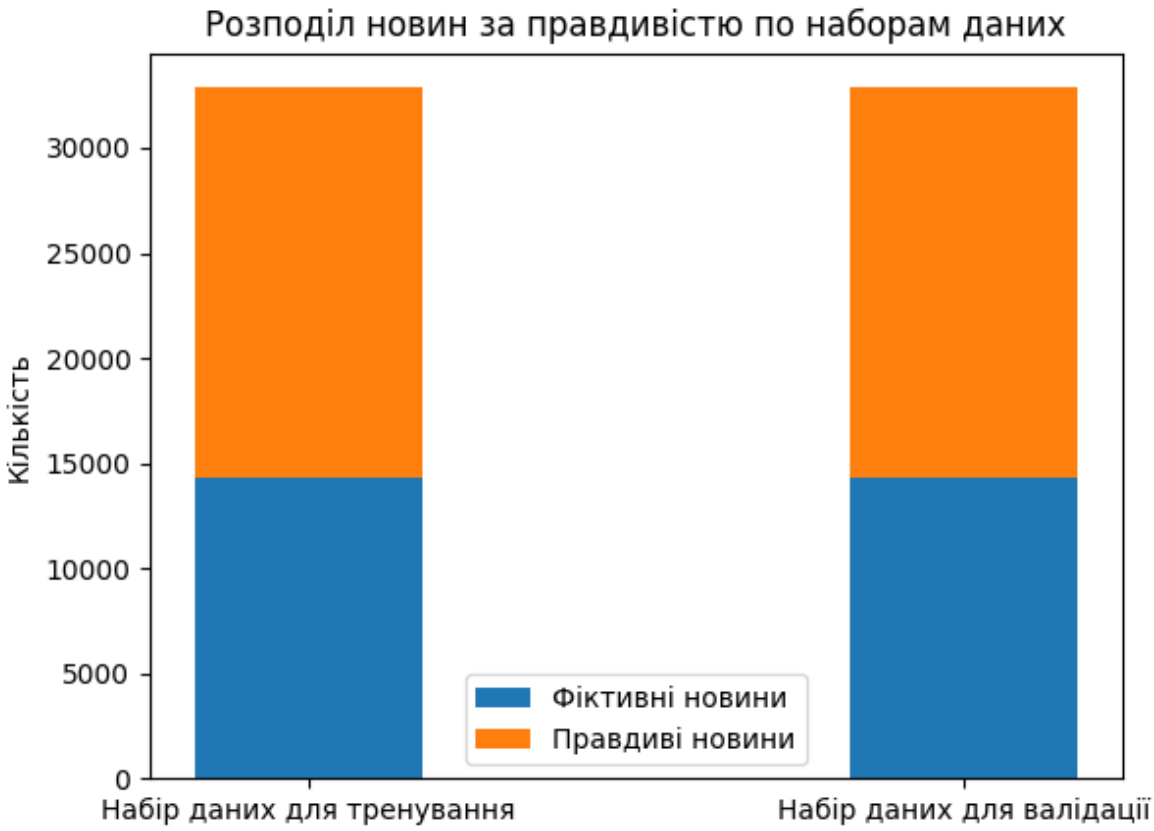


Рис. 4.2. Діаграма розподілу новин за правдивістю по наборам даних

Попри спробу забезпечення різноплановості новин шляхом злиття фіктивних новин про життя зірок з політичними новинами, кінцевий композитний набір даних містить переважно політичні новини.

Це демонструють представлені візуально зважені списки слів, що зустрічаються у тексті новин (рис. 4.3, 4.4).

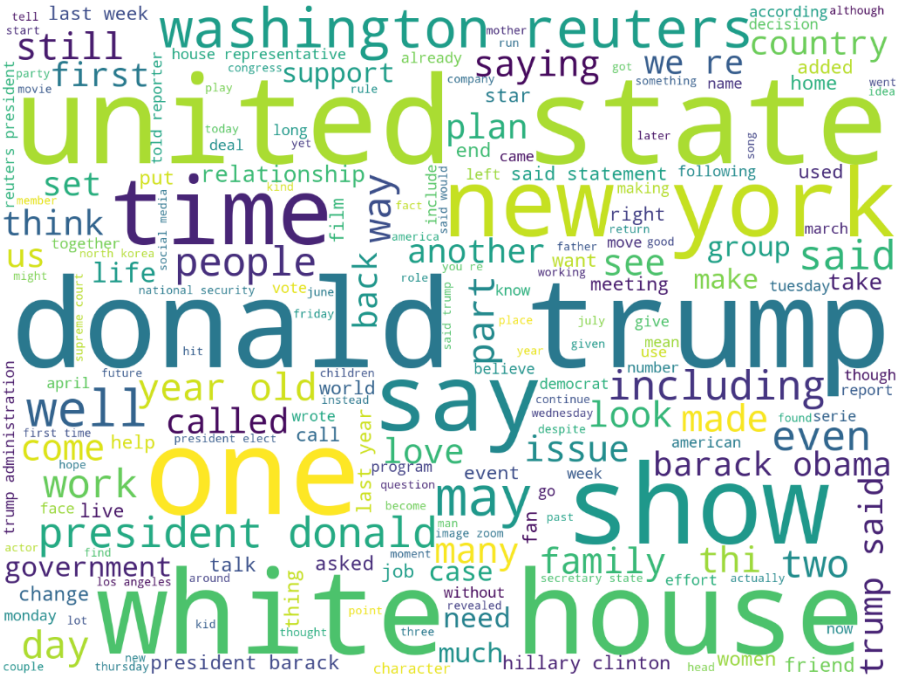


Рис. 4.3. Представлений візуально зважений список слів, що зустрічаються у тексті правдивих новин

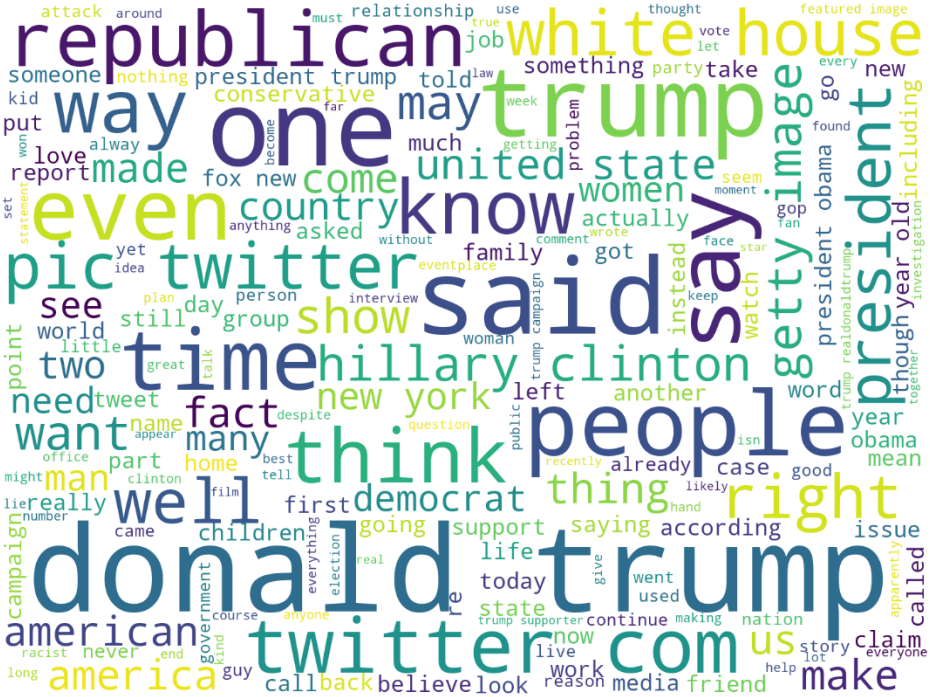


Рис. 4.4. Представлений візуально зважений список слів, що зустрічаються у тексті фіктивних новин

4.3. Реалізація алгоритму класифікації

Для реалізації алгоритму класифікації використано бібліотеку Tensorflow [38]. Розроблена командою Google Brain Team, бібліотека TensorFlow може бути використана для різноманітних завдань щодо оброблення природної мови та проведення складних досліджень з використанням машинного навчання та глибоких нейронних мереж.

4.3.1. Попередня обробка даних

Перед тим, як почати побудову і навчання моделі, дані, отримані за запитом з БД проходять попередню обробку. Даний крок необхідний для того, що прибрати беззмістовну інформацію, що не буде впливати на класифікацію. Всі слова приводяться до нижнього регістру, відбувається видалення стоп-слів, посилань, HTML-тегів.

4.3.2. Векторизація набору даних

Для даного кроку алгоритму потрібно здійснити перехід від тексту як набору слів до його чисельної репрезентації. Для підвищення точності також використано розкладання слова на підслова (англ. Subwords): оскільки значення слів не змінюються від форми чи відмінку у який вони поставлені, то виділення основної частини слова і аналіз частотності його входження в корпус тексту сприяє більш точним результатам класифікації.

Для цього використано SubwordTextEncoder – клас бібліотеки tensorflow_datasets [39], що виконує розкладання слова на фрагменти та закодовує їх. Кодування є повністю зворотним, оскільки всі слова, які не входять до кінцевого словника, кодуються байтами. Розділові знаки видаляються.

Розмір словника задається параметром num_words, що репрезентує максимальну кількість слів, які будуть збережені, залежно від частоти їх вживання в тексті. В моделі, що описується, розмір словника становить 20000 слів.

Після цього відбувається приведення усієї текстової інформації до загального вигляду методом pad_sequences класу keras.preprocessing.sequence. Keras – інтерфейс високого рівня побудований на базі бібліотеки Tensorflow [40]. Метод pad_sequences забезпечує приведення розміру векторів корпусів тексту до однієї довжини шляхом доповнення вектору нулями, за умови недостатньої довжини, або скорочення розміру вектора шляхом видалення зайвих змінних. У даній роботі довжина векторів текстової інформації становить 120 слів.

4.3.3. Структура моделі класифікатора

Функція втрат або функція витрат – це функція, яка відображує подію, або значення однієї чи декількох величин, на дійсне число, яке інтуїтивно представляє якісь «витрати», пов'язані з цією подією. У даній моделі у якості функції витрат була використана функція бінарної кросентропії. Сфера використання функції – проблеми класифікації, в яких рішення є бінарним: так або ні; 1 або 0. Втрата вказує на те, наскільки помилкові прогнози моделі.

Отже, підвищення точності моделі напряму пов’язане зі зменшенням показника втрат. На мінімізацію функції втрат напрямлена діяльність функції оптимізації. У даній роботі використовується оптимізатор, що реалізує алгоритм Адама. Оптимізація Адама – це метод стохастичного градієнтного спуску, який базується на адаптивній оцінці моментів першого і другого порядку.

Keras також забезпечує функціонал побудови Послідовних моделей (англ. The Sequential model) машинного навчання.

Використання послідовної моделі є доцільним у контексті даної роботи, адже оптимальними умовами для її використання є наявність простого стеку шарів, де кожен шар має рівно один вхідний тензор і один вихідний тензор.

Модель, що було використано у даній роботі, утворено чотирма шарами (рис. 4.7):

1. Embedding layer – це поняття відображення дискретних об'єктів, таких як слова, до векторів та реальних чисел. Даний етап є важливим для переходу від тексту до його чисельної репрезентації, яку можна обробляти подальшим алгоритмом методами машинного навчання. Концепція включає стандартні функції, які ефективно перетворюють дискретні вхідні об'єкти у корисні вектори. Розмір словника, сформованого найчастіше зустрічающимися словами із двох датасеті – 20000, розмірність результуючого вектора – 16, константна довжина вхідних послідовностей – 12.
2. Flatten layer – згладжує вхідні дані. Даний шар необхідний для того, щоб привести дані до вигляду, у якому їх можна буде пропустити через наступний рівень.
3. Dense layer – це глибоко зв'язаний шар нейронної мережі, найпоширеніший і часто використовуваний шар. Dense шар виконує наступну операцію на вході та повертає вихідний результат:

output = activation(dot(input, kernel) + bias)

У даному випадку кількість вихідних точок – 6, а функція активації – ReLU (рис. 4.5). ReLU (англ. Rectified Linear Unit) – ReLU – це найпоширеніша в функція активації.

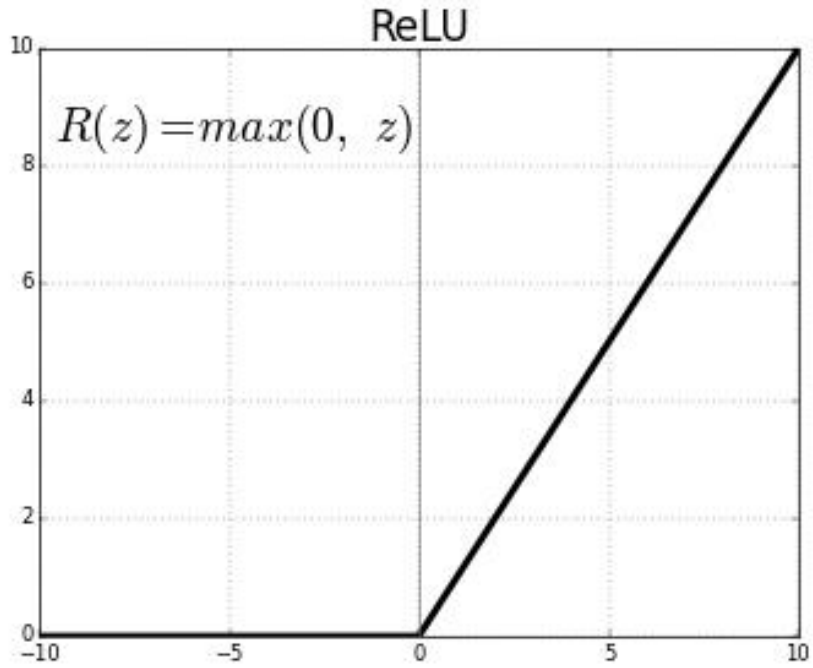


Рис. 4.5. Графік функції ReLU

F (z) дорівнює нулю, при значенні z менше нуля, f (z) дорівнює z, коли z більше або дорівнює нулю.

Діапазон функції: [0 до inf)

4. Dense layer – у цьому шарі кількість вихідних точок – 1, а за функцію активації взята функція sigmoid (рис. 4.6).

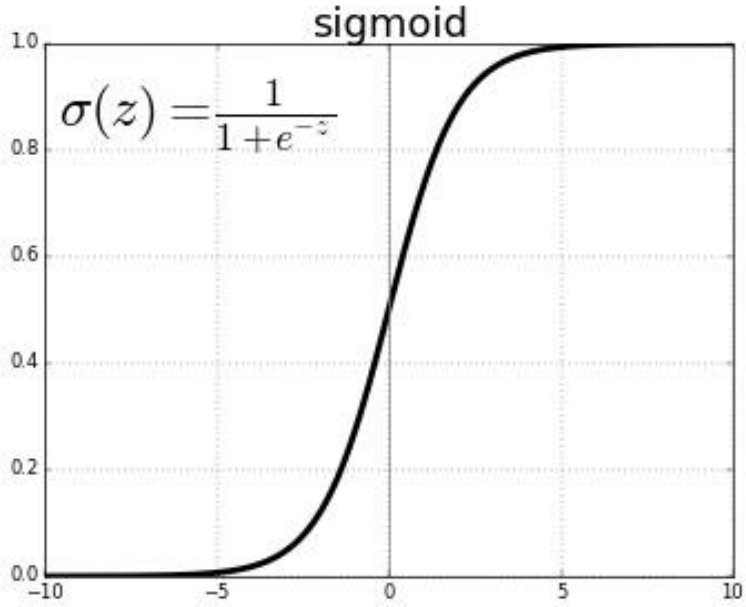


Рис. 4.6. Графік сигмоподібної функції

Основна причина, використання сигмоподібної функції, полягає в тому, що вона існує в діапазоні [0, 1]. Відповідно, якщо вихідне значення ближче до 0, то новина вважається фіктивною, або правдивою у протилежному випадку.

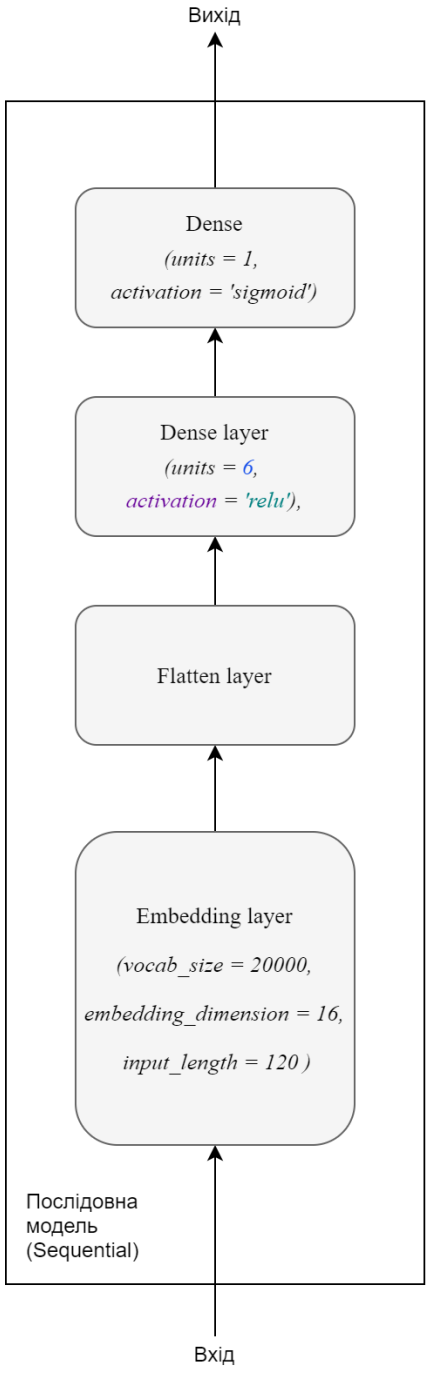


Рис. 4.7. Послідовна модель із переліком шарів, що була використана для виконання класифікації новин

Після навчання, модель і оптимізатор зберігаються в HDF5-файл який містить їх стан (ваги і змінні) і конфігурацію моделі. Експортована модель надалі буде використана із серверної частини системи без доступу до вихідного коду Python.

4.4. Альтернативні реалізації алгоритму класифікації

У підрозділі 4.3 даного проєкту було розглянуто модель класифікації, що була використана у кінцевій реалізації модуля класифікації текстових новин. Проте в ході розроблення алгоритму класифікації було розглянуто декілька послідовних моделей, що відрізнялися методом векторизації блоків тексту або використовуваними шарами. Розглянемо дані варіації моделей класифікатора.

4.4.1. Чотирьохшарова модель з використанням токенізатору

Дана модель містить ті самі шари, що і кінцева модель, проте для векторизації корпусу тексту використовує клас Tokenizer інтерфейсу Keras. Даний клас дозволяє векторизувати текстовий корпус, перетворюючи кожну текстову одиницю (речення, блок тексту тощо) або на послідовність цілих чисел (кожне ціле число є індексом лексеми у словнику), або у вектор, де коефіцієнт для кожного токену може бути двійковим, на основі кількості слів або заснованим на tf-idf. При цьому спочатку всі розділові знаки видаляються, перетворюючи текст в розділені пробілами послідовності слів (слова можуть містити символ '). Після цього ці послідовності розбиваються на списки лексем. А вже потім відбувається індексація або векторизація. Розмір словника в даній моделі становить 10000 слів, а довжина векторів текстової інформації – 120 слів.

4.4.2. Трьохшарова модель з використанням двонапрямленої рекурентної нейронної мережі

Дана модель здійснює векторизацію методом, описаним у підрозділі 4.3.2 даного дипломного проєкту, проте відрізняється структурою моделі класифікатора (рис 4.8).

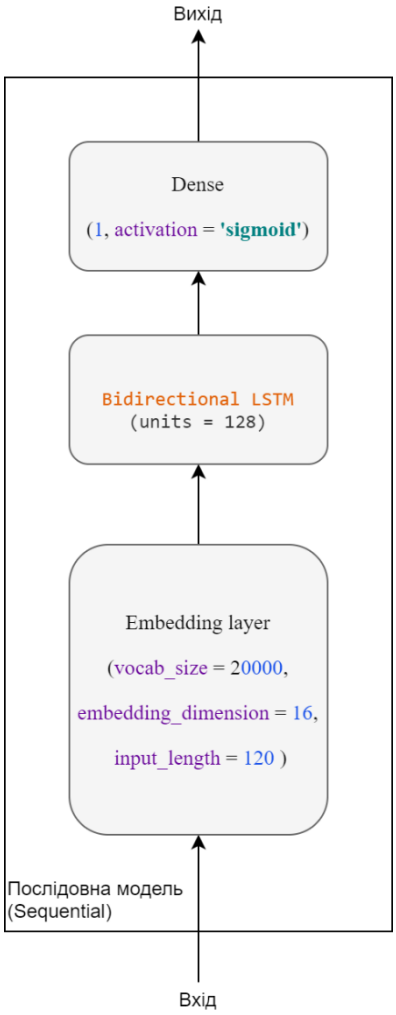


Рис. 4.8. Послідовна трьохшарова модель з використанням двонапрямленої рекурентної нейронної мережі

«Двонаправлена рекурентна нейронна мережа (англ. bidirectional recurrent neural network, BRNN) з'єднує у протилежних напрямках два прихованих шари з однаковим входом» [41]. Таким чином, при формуванні вихідного шару мережі враховується зміст її попереднього та майбутнього

станів, завдяки чому аналіз тексту відбувається не лише на рівні частотності входження слова в корпус тексту певного характеру (фіктивної або правдивої новини), а й із розумінням контексту, в якому слово було використано. Розмірність вихідного вектору двонапрямленної RNN (англ. Recurrent Neural Network) – 128.

Перший і третій шари повністю співпадають з шарами, що використані у кінцевій моделі, їх детальний опис приведено в підрозділі 4.3.3.

4.4.3. П’ятишарова модель з використанням згорткової мережі

Дана модель також використовує для векторизації метод, що описано в підрозділі 4.3.2, та шари Embedding і Dense, опис яких приведено в підрозділі 4.3.3.

При побудові моделі класифікатора використовується згорткова нейронна мережа (англ. CNN). Згорткова нейронна мережа – це реалізація регуляризованої версії багатошарових персептронів, тобто повністю пов’язаних мереж, в яких кожен нейрон в одному шарі з’єднаний з усіма нейронами в наступному шарі.

Згортковий шар Conv1D створює ядро згортки, яке поєднується з входом шару за допомогою одного просторового (або тимчасового) виміру, щоб створити вихідний тензор.

Наступний за згортковим – агрегувальний шар GlobalAveragePooling1D. Його задача – об’єднати групи вихідних нейронів попереднього шару в одиничні нейрони для наступного шару. Алгоритм об’єднання визначається функцією Загального усереднювального агрегування: це значить, що вихідні нейрони формуються використовуючи усереднене значення нейронів попереднього шару.

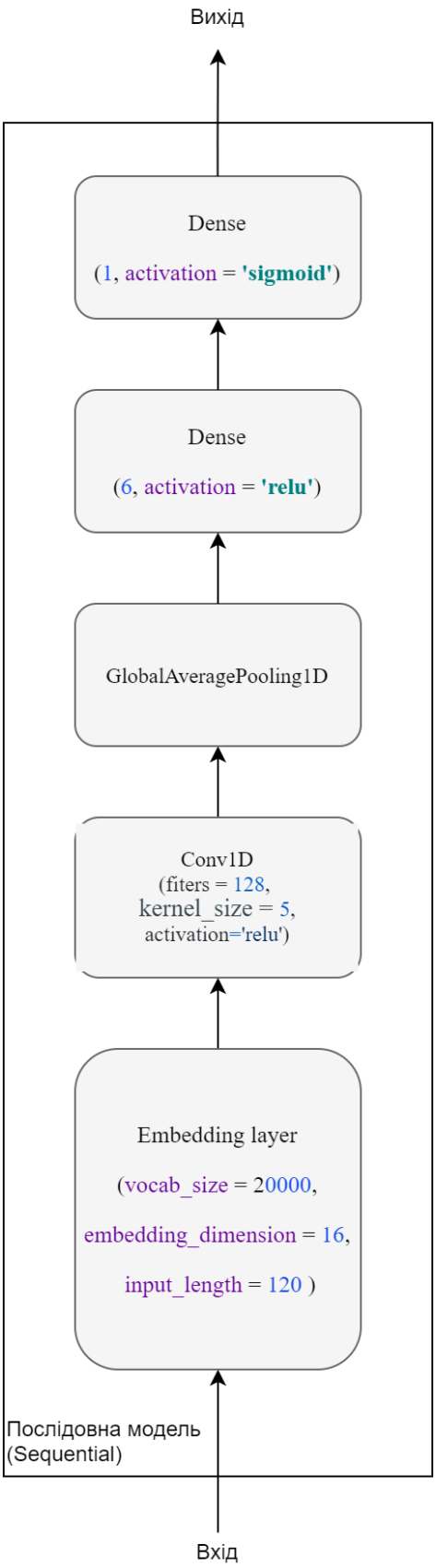


Рис. 4.9. Послідовна п’ятишарова модель з використанням згорткового шару

4.4.4. Обґрунтування кінцевого вибору

В результаті аналізу усіх наведених моделей класифікації текстових новин (табл. 4.1), обрана була чотирьохшарова послідовна модель, що використовує алгоритм векторизації підслів, оскільки саме вона продемонструвала найвищий показник точності, що становить 93,15%.

Таблиця 4.1

Порівняння моделей класифікації текстових новин

Назва моделі	Точність класифікації, %
Чотирьохшарова послідовна модель, що використовує алгоритм векторизації підслів	93,15
Чотирьохшарова послідовна модель, що використовує алгоритм токенізації слів	91,97
Трьохшарова модель з використанням двонапрямленої рекурентної нейронної мережі	92,68
П’ятишарова модель з використанням згорткової мережі	91,88

4.5. Реалізація сервісу ранкування сайту-джерела новини

API забезпечує надання рейтингу сайту відповідно переліку сайтів, що публікували фіктивні новини, або іншу підозрілу інформацію.

Джерело рейтингу сайтів – politifact.com (джерело також завжди вказано в нижньому колонтитулі сторінки користувацького інтерфейсу, рис. 4.10)

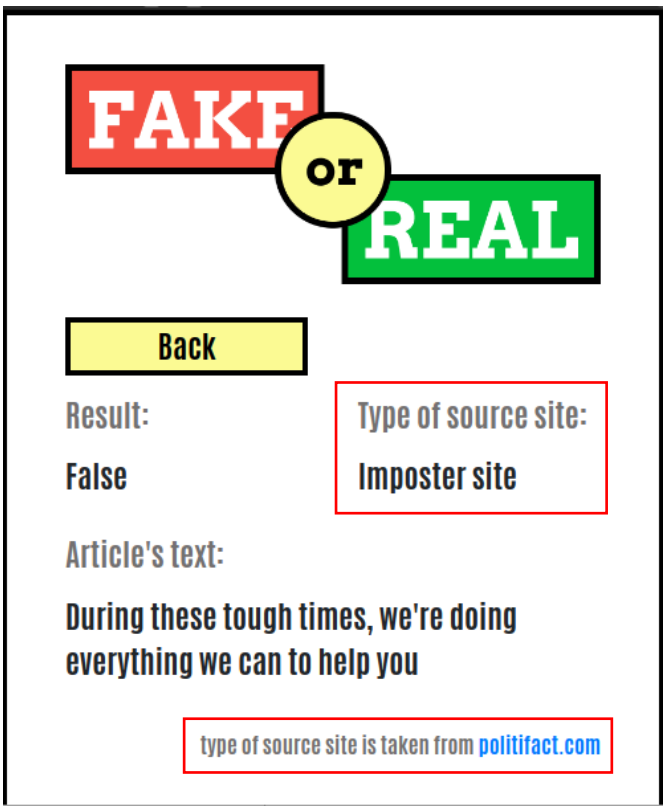


Рис. 4.10. Відображення рейтингової оцінки сайту у інтерфейсі користувача

Можливі типи сайтів за класифікацією politifact.com, що приводяться в пункті «Type of source site» сторінки результатів:

- not listed – означає, що даний сайт відсутній у переліку неправдивих сайтів;
- fake news – сайт, що використовується для публікації фіктивних новин;
- some fake stories – сайт, на якому фіктивні новини публікувалися на рівні з правдивими;
- imposter site – сайт, що імітує інтерфейс сайту іншого Інтернет видання, та знаходиться на доменній адресі, що схожа на доменну адресу оригінального сайту;
- parody site – пародійний сайт; від попереднього типу відрізняється тим, що публікації на даному сайті зроблені із сатиричною метою.

4.6. Реалізація інтерфейсу користувача

Інтерфейс користувача є мінімалістичним та інтуїтивно зрозумілим. Для активації розширення, потрібно натиснути на його піктограму у навігаційній панелі браузера. Основний екран містить дві кнопки: “Paste Selection”, що використовується для вставки виділеного на сторінці тексту у поле вводу (текст також можна вставити або набрати вручну) та “Check” для виконання аналізу вставленого тексту новини (рис. 4.11).

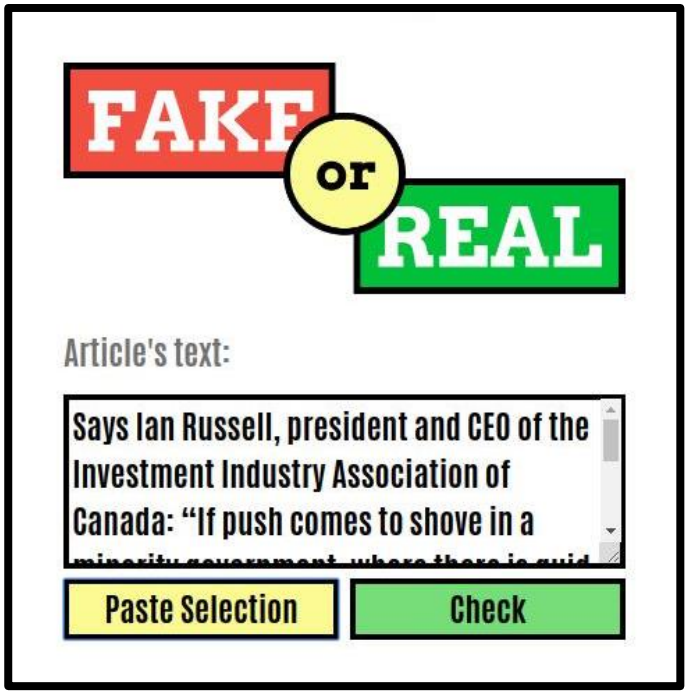


Рис. 4.11. Основний екран розширення

На другому екрані розширення користувач бачить результат аналізу тексту новини під словом “Result”, рейтинг веб-сайту джерела (див. підрозділ 4.5) та сам текст аналізованої новини.

Після перегляду результату аналізу користувач може перейти до попереднього екрану, натиснувши кнопку “Back” або вийти із розширення, натиснувши будь-де поза межами вікна розширення.

4.7. Особливості визначення точності моделі класифікації

Нейронні мережі використовують принцип випадковості, щоб гарантувати, що вони ефективно досліджують функцію, яка апроксимується. Випадковість використовується тому, що цей клас алгоритму машинного навчання працює з ним краще, ніж без.

Найпоширенішою формою випадковості, що використовується в нейронних мережах, є випадкова ініціалізація мережевих ваг (англ. weights). Хоча випадковість може бути використана і в таких областях, як:

- ініціалізація;
- регуляризація;
- побудова шарів;
- функції оптимізації.

Наслідком наявності даних джерел випадковості є те, що запустивши алгоритм нейронної мережі із тими самими вхідними даними, користувач гарантовано отримає різні результати.

Традиційний та практичний спосіб вирішити цю проблему – багато разів запускати свою мережу (30+) та використовувати статистику для підведення підсумків продуктивності вашої моделі та порівняння своєї моделі з іншими моделями.

За результатами 50 вимірів (рис. 4.13), точність отриманої моделі становить 91,664 % (рис. 4.12).

```
Average Validation accuracy for 50 train cases:
0.9314655065536499
```

Рис. 4.12. Середня точність моделі за результатами 50 вимірів

```
Validation accuracy for 50 train cases:
0.935028612613678   0.9320781230926514
0.9313480854034424  0.933903157711029
0.9327168464660645  0.9255383610725403
0.9268767237663269  0.9369449019432068
0.9377357363700867  0.9286105632781982
0.9209758043289185  0.9315609931945801
0.9338727593421936  0.9339335560798645
0.9317739605903625  0.9323214292526245
0.9254167079925537  0.9310439229011536
0.9362452626228333  0.9296143054962158
0.9342073202133179  0.9281846880912781
0.9364886283874512  0.9379791021347046
0.9296751618385315  0.932534396648407
0.9325647950172424  0.936640739440918
0.9230137467384338  0.9335077404975891
0.9336597919464111  0.9235308170318604
0.9358498454093933  0.9276676177978516
0.9341769218444824  0.9261467456817627
0.9264509081840515  0.9375836253166199
0.9353935718536377  0.933081865310669
0.9313480854034424  0.9310135245323181
0.9272722005844116  0.9343594312667847
0.9340552091598511  0.9258729815483093
0.9332643747329712  0.9330514669418335
0.9296143054962158  0.9275459051132202
```

Рис. 4.13. Значення точності тренованої моделі для 50 випадків

ВИСНОВКИ

Метою даного дипломного проєкту є розроблення системи для класифікації новин, а саме, ідентифікації новини як правдивої або фіктивної та надання рейтингу сайту-джерела новини.

Дослідження наявних аналогів призвело до формулювання переліку вимог до розроблюваної системи, що поєднували б найкращі характеристики проаналізованих систем із специфічними функціональними особливостями в одному програмному продукті.

Аналіз доступних мов програмування та фреймворків вказав на доцільність використання Python та фреймворку Flask для створення серверної частини системи та реалізації клієнтської частини додатку у вигляді розширення для веб-браузера Google Chrome. А процесі проєктування системи було розроблено та описано алгоритм класифікації.

Розроблена система для класифікації текстових новин визначає правдивість новини із точністю у 93,15% за середнім результатом п’ятдесяти вимірювань.

Розроблення було виконано в повному обсязі згідно з положеннями Технічного завдання, проведено тестування у відповідності до затвердженої програми та методик тестування.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Elle Hunt, What is fake news? How to spot it and what you can do to stop it [Електронний ресурс] // The Guardian – 2016. – Режим доступу: https://www.theguardian.com/media/2016/dec/18/what-is-fake-news-pizzagate
2. J. Clement, Worldwide digital population as of April 2020 [Електронний ресурс] // Statista – 2020. – Режим доступу: https://www.statista.com/statistics/617136/digital-population-worldwide/
3. Fake news in 2016: What it is, what it wasn't, how to help [Електронний ресурс] // BBC – 2016. – Режим доступу: https://www.bbc.com/news/world-38168792
4. Julia Hunt, 'Fake news' named Collins Dictionary's official Word of the Year for 2017 [Електронний ресурс] // Independent – 2017. – Режим доступу: https://www.independent.co.uk/news/uk/home-news/fake-news-word-of-the-year-2017-collins-dictionary-donald-trump-kellyanne-conway-antifa-corbynmania-a8032751.html
5. Fake news definition [Електронний ресурс] // Cambridge Dictionary– Режим доступу: https://dictionary.cambridge.org/dictionary/english/fake-news
6. Callum Keown, Facebook, Twitter, Google and other tech giants join forces to fight coronavirus fake news [Електронний ресурс] // MarketWatch – 2020. – Режим доступу: https://www.marketwatch.com/story/facebook-twitter-google-and-other-tech-giants-join-forces-to-fight-coronavirus-fake-news-2020-03-17
7. Kul Bhusnan, WhatsApp bans over 2 million accounts monthly for spam, platform abuse [Електронний ресурс] // Hindustantimes – 2020. – Режим доступу: https://www.hindustantimes.com/tech/whatsapp-bans-over-2-million-accounts-monthly-for-spam-platform-abuse/story-f56IvWtW8hrcnTysqafClL.html

8. Reading Metadata To Combat Disinformation And Fake News Campaigns [Електронний ресурс] // School of information – Режим доступу: https://www.ischool.utexas.edu/news/reading-metadata-combat-disinformation-and-fake-news-campaigns
9. Fatemeh Torabi Asr, Maite Taboada, Big Data and quality data for fake news and misinformation detection [Електронний ресурс] // SAGE journals – 2019. – Режим доступу: https://journals.sagepub.com/doi/full/10.1177/2053951719843310
10. Megan Carey , Off the Record: Combatting Disinformation Through Metadata, 2010-2018 IG Status, Permanent Electronic Records [Електронний ресурс] – 2019. – Режим доступу: https://www.tsl.texas.gov/slrm/blog/2019/03/off-the-record-combatting-disinformation-through-metadata-2010-2018-ig-status-permanent-electronic-records/
11. fake_news_detection (99.48% accuracy) [Електронний ресурс] – 2020. – Режим доступу: https://www.kaggle.com/utsavtalwar/fake-news-detection-99-48-accuracy
12. Fake News Challenge Stage 1 (Fnc-I): Stance Detection [Електронний ресурс] – 2017. – Режим доступу: http://www.fakenewschallenge.org/
13. Yuxi Pan, Doug Sibley, Sean Baird, Fake News Challenge – Team SOLAT IN THE SWEN [Електронний ресурс] – 2017. – Режим доступу: https://github.com/Cisco-Talos/fnc-1
14. Andreas Hanselowski, Avinesh PVS, Benjamin Schiller and Felix Caspelherr, Athene, FNC-1 Submission [Електронний ресурс] – 2017. – Режим доступу: https://github.com/hanselowski/athene_system
15. Benjamin Riedel, Isabelle Augenstein, Georgios Spithourakis, Sebastian Riedel, UCL Machine Reading – FNC-1 Submission [Електронний ресурс] – 2017. – Режим доступу: https://github.com/uclnlp/fakenewschallenge
16. Fact Check Explorer [Електронний ресурс]. Режим доступу: https://toolbox.google.com/factcheck/explorer

17. Server-Side Javascript: Back With a Vengeance: [Електронний ресурс]. Режим доступу: https://readwrite.com/2009/12/17/server-side_javascript_back_with_a_vengeance/
18. StackOverflow Most Popular Technologies: [Електронний ресурс]. Режим доступу: https://insights.stackoverflow.com/survey/2019/#technology
19. A simple guide to load C/C++ code into Node.js JavaScript Applications: [Електронний ресурс]. Режим доступу: https://itnext.io/a-simple-guide-to-load-c-c-code-into-node-js-javascript-applications-3fcccf54fd32
20. Chapter 4. How JavaScript Was Created: [Електронний ресурс]. Режим доступу: http://speakingjs.com/es5/ch04.html
21. Top 12 JavaScript Machine Learning Libraries for 2020: [Електронний ресурс]. Режим доступу: https://medium.com/javascript-in-plain-english/top-12-javascript-machine-learning-libraries-for-2020-8bec30ecfa58
22. Programming languages used in most popular websites: [Електронний ресурс]. Режим доступу: https://www.wikiwand.com/en/Programming_languages_used_in_most_popular_websites
23. Program Java devices -- An overview: [Електронний ресурс]. Режим доступу: https://www.javaworld.com/article/2076441/java-se-program-java-devices-an-overview.html
24. Machine Learning Open Source Software: [Електронний ресурс]. Режим доступу: https://mloss.org/software/language/java/?page=1
25. C# Language Specification (PDF) (4th ed.). Ecma International. June 2006. Retrieved January 26, 2012. : [Електронний ресурс]. Режим доступу: https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-334.pdf
26. Xamarin Overview: [Електронний ресурс]. Режим доступу: https://dotnet.microsoft.com/apps/xamarin

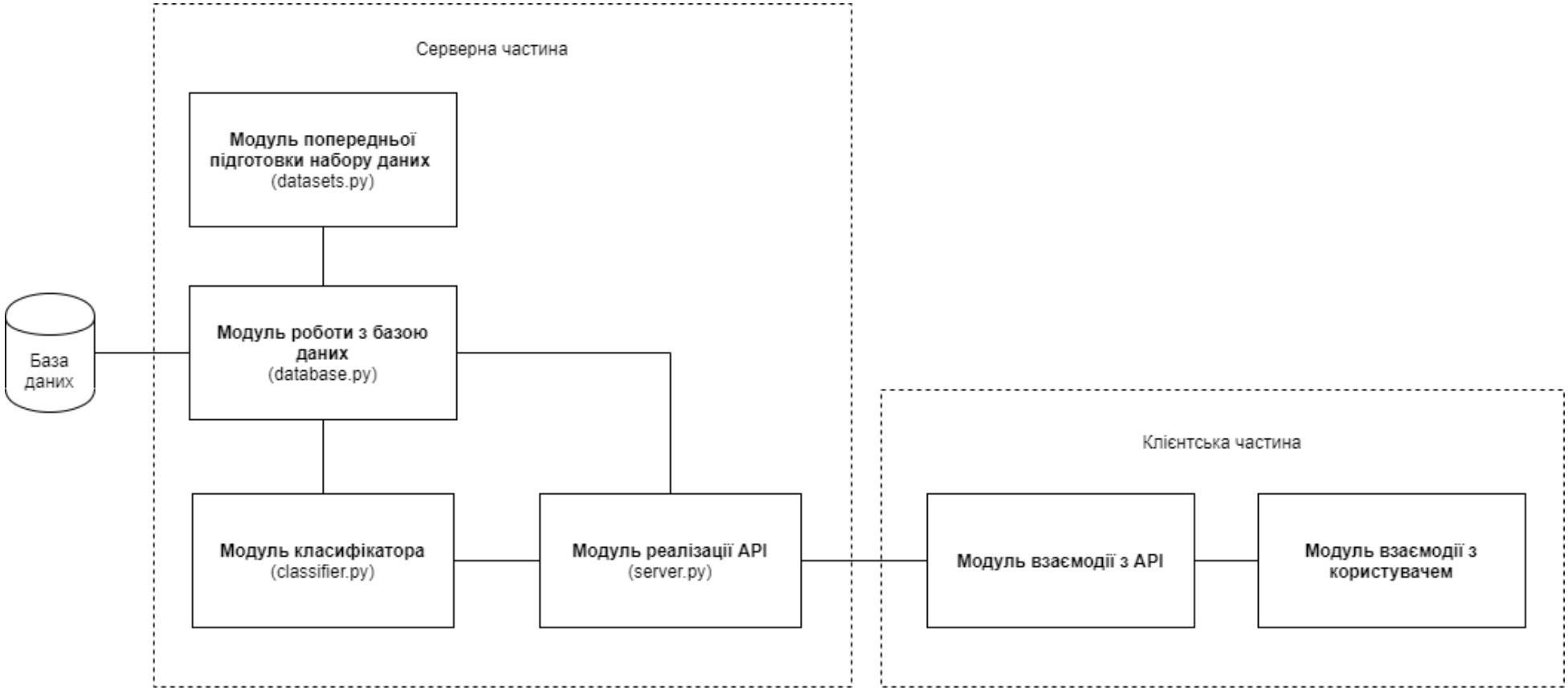
27. ASP.NET features: [Електронний ресурс]. Режим доступу: https://dotnet.microsoft.com/apps/aspnet
28. ASP.NET Core 3.0 Updates: [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/whats-new/dotnet-core-3-0
29. ML libraries for .NET: [Електронний ресурс]. Режим доступу: https://github.com/quozd/awesome-dotnet#machine-learning-and-data-science
30. Chen, Haiping (December 11, 2019). "TensorFlow.NET: .NET Standard bindings for TensorFlow". Retrieved December 11, 2019.: [Електронний ресурс]. Режим доступу: https://github.com/SciSharp/TensorFlow.NET
31. Icaza, Miguel de (February 17, 2019). "TensorFlowSharp: TensorFlow API for .NET languages". Retrieved February 18, 2019. .: [Електронний ресурс]. Режим доступу: https://github.com/migueldeicaza/TensorFlowSharp
32. Stackoverflow Developer Survey Results: [Електронний ресурс]. Режим доступу: https://insights.stackoverflow.com/survey/2020
33. Snigdha Patel, What is Chatbot? Why are Chatbots Important? [Електронний ресурс] // REVE chat – 2020. – Режим доступу: https://www.revechat.com/blog/what-is-a-chatbot/
34. Usage share of web browsers: [Електронний ресурс]. Режим доступу: https://www.wikiwand.com/en/Usage_share_of_web_browsers
35. John Parker, 5 Common Requirement Mistakes [Електронний ресурс] // ENFOCUS solutions – 2011. – Режим доступу: https://enfocussolutions.com/5-common-requirement-mistakes/
36. Fake and real news dataset [Електронний ресурс]. – 2020. – Режим доступу: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
37. KaiDMML/FakeNewsNet [Електронний ресурс]. – 2019. – Режим доступу: https://github.com/KaiDMML/FakeNewsNet
38. TensorFlow is an end-to-end open source platform for machine learning [Електронний ресурс]. Режим доступу: https://www.tensorflow.org/overview

39. TensorFlow Datasets: a collection of ready-to-use datasets [Електронний ресурс]. Режим доступу: https://www.tensorflow.org/datasets/api_docs/python/tfds/features/text/SubwordTextEncoder
40. Keras overview [Електронний ресурс]. Режим доступу: https://www.tensorflow.org/guide/keras/overview
41. Двонаправлені рекурентні нейронні мережі [Електронний ресурс]. Режим доступу: https://www.wikiwand.com/uk/%D0%94%D0%B2%D0%BE%D0%BD%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D1%96_%D1%80%D0%B5%D0%BA%D1%83%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D1%96_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%96_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D1%96

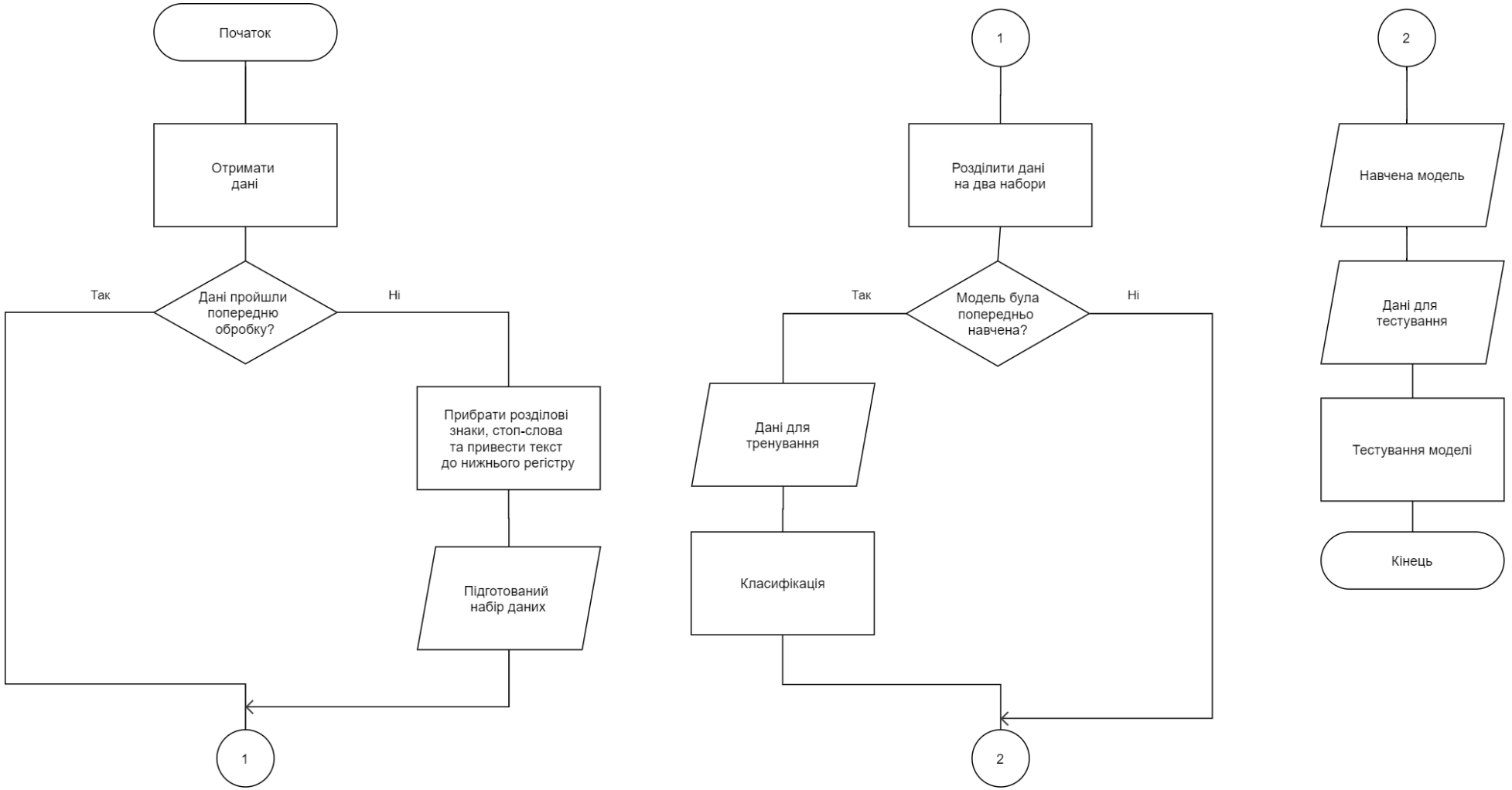
ДОДАТКИ

Додаток 1

Копії графічних матеріалів

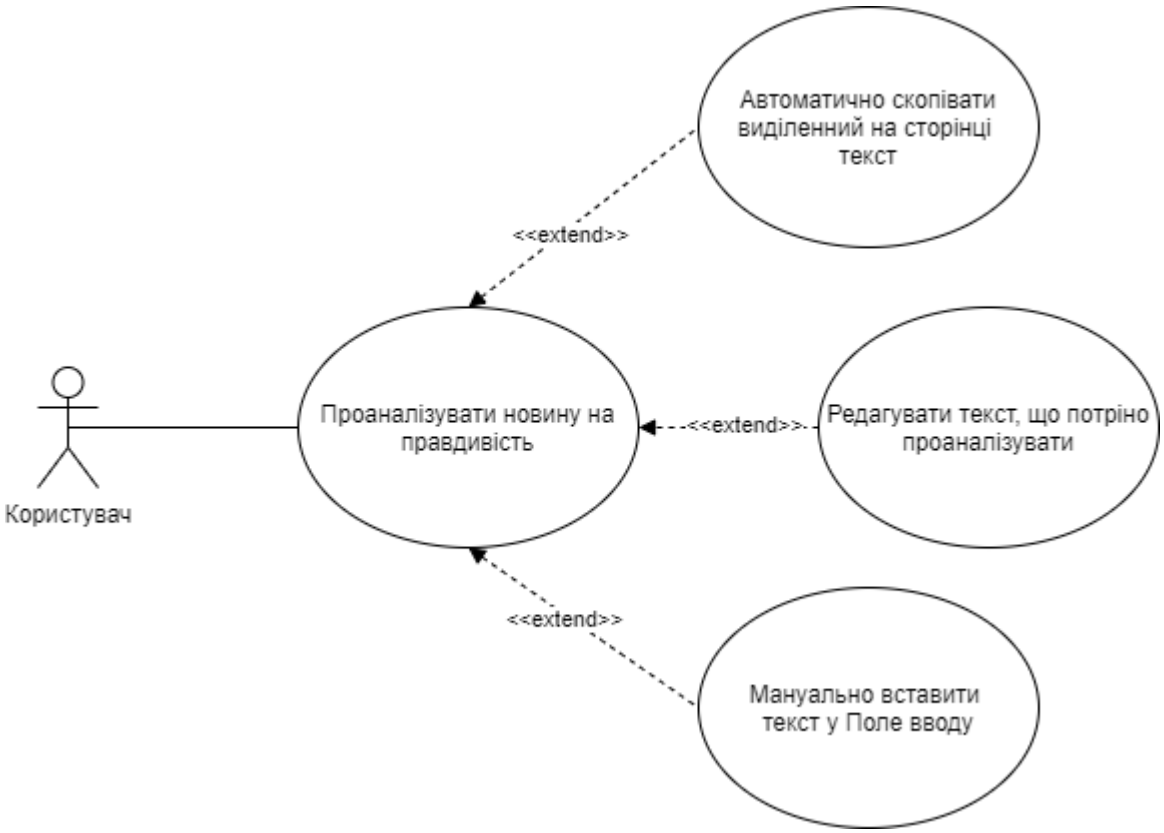


ДП.045440-06-99. Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях. Загальна архітектура системи. UML-діаграма



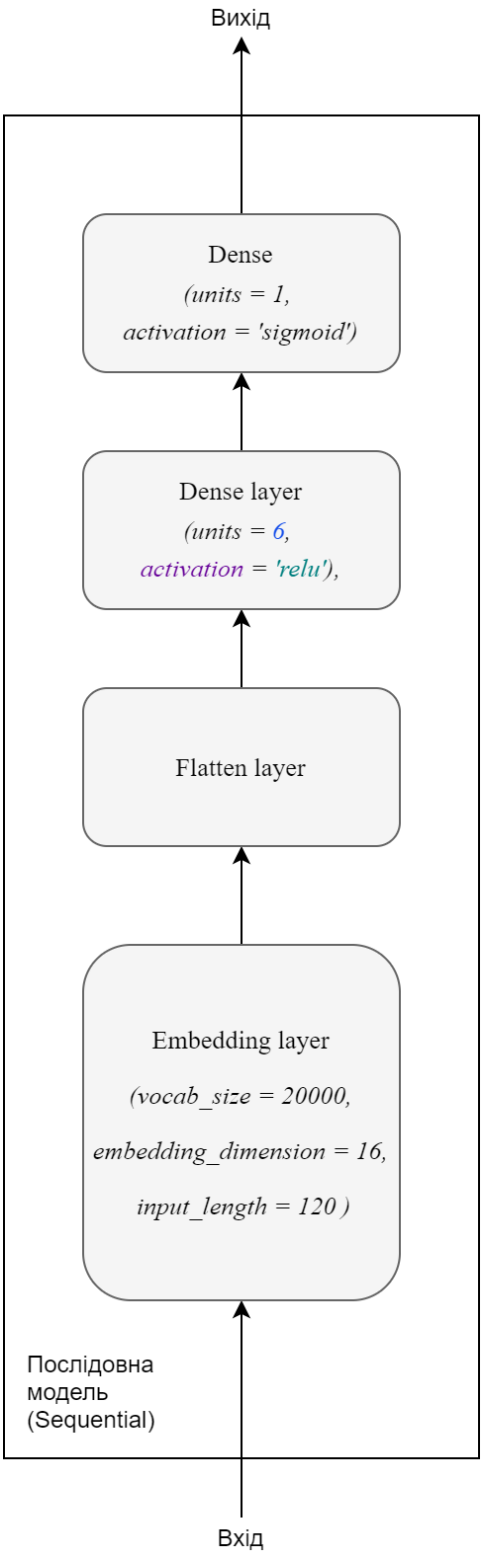
ДП.045440-07-99. Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях. Навчання класифікатора. Схема алгоритму

ДІАГРАМА ПРЕЦЕДЕНТІВ ВЕБ-ДОДАТКУ ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ФІКТИВНИХ ТЕКСТОВИХ НОВИН У ІНТЕРНЕТ-ВИДАННЯХ



Ільїн М.О., група КП-61

СХЕМА ПОТОКУ НАВЧАННЯ МОДЕЛІ ВЕБ-ДОДАТКУ ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ФІКТИВНИХ ТЕКСТОВИХ НОВИН У ІНТЕРНЕТ-ВИДАННЯХ



Ільїн М.О., група КП-61

Додаток 2

Лістинг модулів системи

Лістинг модуля datasets_utils.py

```
import os
import csv
import json
import numpy as np

#General functions
def split_list(a_list):
    half = len(a_list)//2
    return a_list[:half], a_list[half:]

def get_labels(fake_news_quantity, real_news_quantity):
    return np.concatenate((np.full((1, fake_news_quantity), 0, dtype=int),
np.full((1, real_news_quantity), 1, dtype=int)), axis=1)[0]

def remove_stopwords(text):
    stopwords = [ "\n", "a", "about", "above", "after", "again", "against",
"all", "am", "an", "and", "any", "are", "as", "at", "be", "because", "been",
"before", "being", "below", "between", "both", "but", "by", "could", "did",
"do", "does", "doing", "down", "during", "each", "few", "for", "from",
"further", "had", "has", "have", "having", "he", "he'd", "he'll", "he's",
"her", "here", "here's", "hers", "herself", "him", "himself", "his", "how",
"how's", "i", "i'd", "i'll", "i'm", "i've", "if", "in", "into", "is", "it",
"it's", "its", "itself", "let's", "me", "more", "most", "my", "myself",
"nor", "of", "on", "once", "only", "or", "other", "ought", "our", "ours",
"ourselves", "out", "over", "own", "same", "she", "she'd", "she'll", "she's",
"should", "so", "some", "such", "than", "that", "that's", "the", "their",
"theirs", "them", "themselves", "then", "there", "there's", "these", "they",
"they'd", "they'll", "they're", "they've", "this", "those", "through", "to",
"too", "under", "until", "up", "very", "was", "we", "we'd", "we'll", "we're",
"we've", "were", "what", "what's", "when", "when's", "where", "where's",
"which", "while", "who", "who's", "whom", "why", "why's", "with", "would",
"you", "you'd", "you'll", "you're", "you've", "your", "yours", "yourself",
"yourselves" ]
    for word in stopwords:
        token = " " + word + " "
        text = text.replace(token, " ")
        text = text.replace("  ", " ")
    return text
#Functions for working with fake_and_real_0 dataset
class Dataset_0:
    #Returns all news from file
    def get_news_from_file(self, file_path, data_preprocessing):
        news = []
        with open(file_path, 'r') as csvfile:
            reader = csv.reader(csvfile, delimiter=',')
            next(reader)
            for row in reader:
                sentence = row[1]
                # removing stopwords is optional
                if data_preprocessing == "rm_stopwords":
                    remove_stopwords(sentence)
                news.append(sentence)
        return news

    # Returns training and testing datasets with their labels
    def get_all_news(self, data_preprocessing):
        fake_news =
self.get_news_from_file("../datasets/fake_and_real_0/Fake.csv",
data_preprocessing)
```

```
        real_news =
self.get_news_from_file("../datasets/fake_and_real_0/True.csv",
data_preprocessing)

        fn_training, fn_test = split_list(fake_news)
        rn_training, rn_test = split_list(real_news)

        training_ds = fn_training + rn_training
        testing_ds = fn_test + rn_test
        # creating an array of labels that consists is 0 for fake news and 1
for real news
        # result is numpy array
        training_labels = get_labels(len(fn_training), len(rn_training))
        testing_labels = get_labels(len(fn_test), len(rn_test))

        return training_ds, training_labels, testing_ds, testing_labels

#Functions for working with fake_and_real_1 dataset

class Dataset_1:
    def get_news_texts(self, path, data_preprocessing):
        news_list = []
        for dirpath, dirnames, files in os.walk(path):
            for file_name in files:
                if file_name.endswith(".json"):
                    with open(dirpath + '/' + file_name, 'r') as f:
                        file_content = json.load(f)
                        is_fake = 0 if "real" in dirpath else 1
                        if file_content["text"] != '':
                            if data_preprocessing == "rm_stopwords":

news_list.append(remove_stopwords(file_content["text"]))
                            else :
                                news_list.append(file_content["text"])
        return news_list

    def get_all_news(self, data_preprocessing):

        fake_1 =
self.get_news_texts("../datasets/fake_and_real_1/code/fakenewsnet_dataset/gos
sipcop/fake", data_preprocessing)
        real_1 =
self.get_news_texts("../datasets/fake_and_real_1/code/fakenewsnet_dataset/gos
sipcop/real", data_preprocessing)
        fake_2 =
self.get_news_texts("../datasets/fake_and_real_1/code/fakenewsnet_dataset/pol
itifact/fake", data_preprocessing)
        real_2 =
self.get_news_texts("../datasets/fake_and_real_1/code/fakenewsnet_dataset/pol
itifact/real", data_preprocessing)

        f1_training, f1_test = split_list(fake_1)
        r1_training, r1_test = split_list(real_1)

        f2_training, f2_test = split_list(fake_2)
        r2_training, r2_test = split_list(real_2)

        training_ds = f1_training + f2_training + r1_training + r2_training
        testing_ds = f1_test + f2_test + r1_test + r2_test

        training_labels = get_labels(len(f1_training) + len(f2_training),
                                     len(r1_training) + len(r2_training))
        testing_labels = get_labels(len(f1_test) + len(f2_test),
```

```
                                    len(r1_test) + len(r2_test))

        return training_ds, training_labels, testing_ds, testing_labels

class Dataset_Mixed():

    # data_preprocessing - flag that used to determine, whether stopwords
should be removed
    # if equal "rm_stopwords" - stopwords would be removed
    def get_mixed_dataset(self, data_preprocessing):
        ds_0 = Dataset_0()
        ds_1 = Dataset_1()
        training_0, training_labels_0, test_0, test_labels_0 =
ds_0.get_all_news(data_preprocessing)
        training_1, training_labels_1, test_1, test_labels_1 =
ds_1.get_all_news(data_preprocessing)
        merged_training_labels = np.concatenate((training_labels_0,
training_labels_1), axis=0)
        merged_test_labels = np.concatenate((test_labels_0, test_labels_1),
axis=0)

        return training_0 + training_1, merged_training_labels, test_0 +
test_1, merged_test_labels

class Dataset_Sites:
    # Returns all sites
    def get_all_sites():
        sites = []
        with open("../datasets/politifact_list_of_sites/data.csv", 'r') as
csvfile:
            reader = csv.reader(csvfile, delimiter=',')
            next(reader)
            for row in reader:
                sites.append({
                    "type":row[1],
                    "domain":row[0],
                    "source":"politifact.com",

"source_link":"https://www.politifact.com/article/2017/apr/20/politifacts-
guide-fake-news-websites-and-what-they/"
                })
        return sites

        return sites
```

Лістинг модуля database_utils.py

```
# getting MONGO_URI from .env variables
from dotenv import load_dotenv
load_dotenv()
import os
MONGO_URI = os.getenv("MONGO_URI")

import json

# getting dataset from ds_utils
import sys
#sys.path.append("..")
#from datasets_utils.datasets_utils import Dataset_Mixed
```

```
#dsm = Dataset_Mixed()
#training_ds, training_labels, testing_ds, testing_labels =
dsm.get_mixed_dataset("")
from mongoengine import *

# creating Mongo Documents classes
class Article(Document):
    text = StringField(required=True)
    label = IntField(required=True)
    dataset_name = StringField(required=True, max_length=50)
    purpose = StringField(required=True, max_length=50)
    meta = {'collection': 'News'}

class Site(Document):
    type = StringField(required=True)
    domain = StringField(required=True, unique=True)
    source = StringField(required=True)
    source_link = StringField(required=True)
    meta = {'collection': 'Sites'}

class Word(Document):
    vec = ListField(required=True)
    word = StringField(required=True, unique=True)
    train_dataset_name = StringField(required=True)
    train_algorithm = StringField(required=True)
    meta = {'collection': 'Dictionary'}

class Database:
    connection = connect('heroku_1gkq932s', host=MONGO_URI +
"?retryWrites=false")
    def insert_site(self, site_document):
        Site(type=site_document["type"],
domain=site_document["domain"].lower(), source=site_document["source"],
source_link=site_document["source_link"]).save()

    def insert_word(self, _vec, _word, _train_dataset_name,
_train_algorithm):
        Word(vec=_vec,
             word=_word,
             train_dataset_name=_train_dataset_name,
             train_algorithm=_train_algorithm).save()

    def insert_article(self, _text, _label, _ds_name, _purpose):
        Article(text=_text,
             label=_label,
             dataset_name=_ds_name,
             purpose = _purpose).save()

    def get_site_info(self, _domain):
        try:
            return Site.objects.get(domain=_domain).to_json()
        except Exception:
            return {
                "type": "Not listed",
                "domain": _domain,
                "source": "none",
                "source_link": "none"
            }

    def get_articles_by_ds_and_purpose(self, _dataset_name, _purpose):
        def blank (data):
            print("Hey yo")
```

```
        try:
            articles = (Article.objects.filter(dataset_name=_dataset_name,
purpose=_purpose).exclude('dataset_name').exclude('purpose'))

            texts = []
            labels = []
            for article in articles:
                texts.append(article.text)
                labels.append(article.label)
            return {"articles":texts, "labels":labels}

        except Exception as e:
            return "error: " + str(e)



#from datasets_utils.datasets_utils import Dataset_Sites

#db = Database()
#sites = Dataset_Sites.get_all_sites()
#print(sites)
#for site in sites:
    #db.insert_site(site)
```

Лістинг модуля server.py

```
from src.database_utils import Database
from src.classifier import Classifier
from flask import Flask, redirect

app = Flask(__name__)

@app.route('/api')
def index():
    return "Fake or Real API<br><br>/api/v1.0/news/classify/<string:article>
- to get article analysis result<br>/api/v1.0/site/type/<string:domain> - to
get site type"

@app.route('/api/v1.0/news/classify/<string:article>', methods=['GET'])
def classify(article):
    classifier = Classifier()
    result = classifier.predict(article)
    print ("RESULT: ")
    print(result)
    if result >= 0.5:
        result = {
            "result": "True"
        }
    else:
        result = {
            "result": "False"
        }

    return result

@app.route('/api/v1.0/site/type/<string:domain>', methods=['GET'])
def get_site_type(domain):
    db = Database()
    return db.get_site_info(domain)
```

```
@app.route('/', methods=['GET'])
def red():
    return redirect("/1", code=302)

@app.route('/<int:page>', methods=['GET'])
def get_index(page):
    db = Database()
    val_news = db.get_articles_by_page("mixed", "test", page)

    import chevron

    args = {
        'template': "./views/index_template.html",
        'data': {
            "news": val_news
        }
    }
    print(val_news)
    with open("./views/index_template.html", 'r') as f:
        return chevron.render(f, {"news": val_news})

    #return chevron.render(**args)


if __name__ == '__main__':
    app.run(debug=True)
```

Лістинг модуля popup.js

```
function pasteSelection(e) {
        chrome.tabs.executeScript( {
                        code: "window.getSelection().toString();"
                }, function(selection) {
                        document.getElementById("text-area").value =
selection[0];
        });
}

function backToPaste(e) {
        document.getElementById("analyzis-screen").style.display = "none";
        document.getElementById("paste-screen").style.display = "inline";
}

function analyze(e) {
        let domain;
        let article = document.getElementById("text-area").value;
        chrome.tabs.query({
                active: true,
                currentWindow: true
        }, ([currentTab]) => {
                const url = new URL(currentTab.url);
                domain = (url.hostname).replace("www.","");

                let api_request_site_type =
'https://fakeorrealnews.herokuapp.com/api/v1.0/site/type/' + domain;
                fetch(api_request_site_type)
                .then((response) => {
                        return response.json();
                })
```

```
                .then((data) => {
                        console.log(data);
                        if (data.type === "Not listed")

        document.getElementById("disclaimer").style.display = "none";
                        else

        document.getElementById("disclaimer").style.display = "block";

                        document.getElementById("type").innerHTML = data.type;
                        document.getElementById("source-site").innerHTML =
data.source;
                        document.getElementById("source-type-list").href =
data.source_link;
                });

        });

        fetch('https://fakeorrealnews.herokuapp.com/api/v1.0/news/classify/' +
article)
        .then((response) => {
                return response.json();
        })
        .then((data) => {
                console.log(data);
                document.getElementById("Result").innerHTML = data.result;
        });




        document.getElementById("article").innerHTML = article;


        document.getElementById("analyzis-screen").style.display = "inline";
        document.getElementById("paste-screen").style.display = "none";
}


document.addEventListener('DOMContentLoaded', function () {
        document.getElementById("analyzis-screen").style.display = "none";
        document.getElementById("paste-screen").style.display = "normal";

        document.querySelector('#paste-button').addEventListener('click',
pasteSelection);
        document.querySelector('#back-button').addEventListener('click',
backToPaste);
        document.querySelector('#check-button').addEventListener('click',
analyze);
});
```

Лістинг модуля classifier.py

```
class Classifier():
    import numpy as np
    import tensorflow as tf
    from tensorflow.keras.preprocessing.text import Tokenizer
    vocab_size = 20000
    embedding_dim = 16
```

```
    max_length = 120
    trunc_type = 'post'
    oov_tok = "<OOV>"
    tokenizer = ""
    training = ""
    validation = ""
    model = ""

    def get_tokenizer(self, name="sample"):
        from tensorflow.keras.preprocessing.text import Tokenizer
        import tensorflow_datasets as tfds
        vocab_filename = name
        # Build
        #tokenizer = tfds.features.text.SubwordTextEncoder.build_from_corpus(
        #    self.training["articles"], target_vocab_size=self.vocab_size)
        #tokenizer.save_to_file(vocab_filename)

        # Load
        encoder =
tfds.features.text.SubwordTextEncoder.load_from_file(vocab_filename)

        #print(encoder.subwords)
        #tokenizer = Tokenizer(num_words=self.vocab_size,
oov_token=self.oov_tok)
        #tokenizer.fit_on_texts(encoder)

        self.tokenizer = encoder

    def remove_stopwords(self, text):
        stopwords = ["\n", "a", "about", "above", "after", "again",
"against", "all", "am", "an", "and", "any", "are",
                     "as", "at", "be", "because", "been", "before", "being",
"below", "between", "both", "but", "by",
                     "could", "did", "do", "does", "doing", "down", "during",
"each", "few", "for", "from", "further",
                     "had", "has", "have", "having", "he", "he'd", "he'll",
"he's", "her", "here", "here's", "hers",
                     "herself", "him", "himself", "his", "how", "how's", "i",
"i'd", "i'll", "i'm", "i've", "if", "in",
                     "into", "is", "it", "it's", "its", "itself", "let's",
"me", "more", "most", "my", "myself", "nor",
                     "of", "on", "once", "only", "or", "other", "ought",
"our", "ours", "ourselves", "out", "over",
                     "own", "same", "she", "she'd", "she'll", "she's",
"should", "so", "some", "such", "than", "that",
                     "that's", "the", "their", "theirs", "them",
"themselves", "then", "there", "there's", "these",
                     "they", "they'd", "they'll", "they're", "they've",
"this", "those", "through", "to", "too",
                     "under", "until", "up", "very", "was", "we", "we'd",
"we'll", "we're", "we've", "were", "what",
                     "what's", "when", "when's", "where", "where's", "which",
"while", "who", "who's", "whom", "why",
                     "why's", "with", "would", "you", "you'd", "you'll",
"you're", "you've", "your", "yours",
                     "yourself", "yourselves"]

        stopwords = [
            "i", "me", "my", "myself", "we", "our", "ours", "ourselves",
"you", "your", "yours", "yourself",
            "yourselves", "he", "him", "his", "himself", "she", "her",
"hers", "herself", "it", "its", "itself", "they",
```

```
            "them", "their", "theirs", "themselves", "what", "which", "who",
"whom", "this", "that", "these", "those",
            "am", "is", "are", "was", "were", "be", "been", "being", "have",
"has", "had", "having", "do", "does",
            "did", "doing", "a", "an", "the", "and", "but", "if", "or",
"because", "as", "until", "while", "of", "at",
            "by", "for", "with", "about", "against", "between", "into",
"through", "during", "before", "after", "above",
            "below", "to", "from", "up", "down", "in", "out", "on", "off",
"over", "under", "again", "further", "then",
            "once", "here", "there", "when", "where", "why", "how", "all",
"any", "both", "each", "few", "more", "most",
            "other", "some", "such", "no", "nor", "not", "only", "own",
"same", "so", "than", "too", "very", "s", "t",
            "can", "will", "just", "don", "should", "now"
        ]
        for word in stopwords:
            token = " " + word + " "
            text = text.replace(token, " ")
            text = text.replace("  ", " ")
        return text

    def get_datasets(self, rm_stopwords = False):
        import sys
        sys.path.append("..")
        from database.database_utils import Database
        import numpy as np

        db = Database()
        training = db.get_articles_by_ds_and_purpose("mixed", "train")
        test = db.get_articles_by_ds_and_purpose("mixed", "test")
        if rm_stopwords == False:
            self.training = {
                "articles": training["articles"],
                "labels": np.array(training["labels"])
            }

            self.validation = {
                "articles": test["articles"],
                "labels": np.array(test["labels"])
            }

        elif rm_stopwords == True:
            clean_ds = []
            for article in training["articles"]:
                clean_ds.append(self.remove_stopwords(article.lower()))
            self.training = {
                "articles": clean_ds,
                "labels": np.array(training["labels"])
            }

            clean_val_ds = []
            for article in test["articles"]:
               clean_val_ds.append(self.remove_stopwords(article.lower()))

            self.validation = {
               "articles": clean_val_ds,
               "labels": np.array(test["labels"])
            }

    def encode_sequences(self):
        from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
        encoded_news = []
        for article in self.training["articles"]:
            encoded_news.append(self.tokenizer.encode(article))

        self.training["padded"] = pad_sequences(encoded_news,
maxlen=self.max_length, truncating=self.trunc_type)

        encoded_val_news = []
        for article in self.validation["articles"]:
            encoded_val_news.append(self.tokenizer.encode(article))

        self.validation["padded"] = pad_sequences(encoded_val_news,
maxlen=self.max_length,
                                                  truncating=self.trunc_type)


    def __init__(self, _tokenizer="local", type=1, rm_stopwords=False):

        if _tokenizer == "local":
            import os, json

            FILE_DIR = os.path.dirname(os.path.abspath(__file__))
            cs_dir = os.path.join(FILE_DIR, 'model')
            tokenizer_file = os.path.join(cs_dir, 'tokenizer.json')
            with open(tokenizer_file) as f:
                import json
                from keras_preprocessing.text import tokenizer_from_json
                data = json.load(f)
                self.tokenizer = tokenizer_from_json(data)

            from pathlib import Path
            path = Path("./model/model.h5")
            path = path.resolve()

            import tensorflow as tf
            self.model = tf.keras.models.load_model(path)

        else:
            self.get_datasets(rm_stopwords)
            if rm_stopwords == True:
                tokenizer =
self.get_tokenizer("model/subwords_vocabulary_20k_rm-words_true")
            else:
                tokenizer =
self.get_tokenizer("model/subwords_vocabulary_20k_rm-words_false")
            self.encode_sequences()
            self.model = self.create_model(type)


    def create_model(self, type = 1):
        import tensorflow as tf

        if type == 0:
            model = tf.keras.Sequential([
                tf.keras.layers.Embedding(self.vocab_size,
self.embedding_dim, input_length=self.max_length),
                tf.keras.layers.Flatten(),
                tf.keras.layers.Dense(6, activation='relu'),
                tf.keras.layers.Dense(1, activation='sigmoid')
            ])

        elif type == 1:
            model = tf.keras.Sequential([
```

```
                tf.keras.layers.Embedding(self.vocab_size,
self.embedding_dim, input_length=self.max_length),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128,
return_sequences=True)),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,
return_sequences=True)),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
                tf.keras.layers.Dense(64, activation='relu'),
                tf.keras.layers.Dense(1, activation='sigmoid')
            ])

        elif type == 2:
            model = tf.keras.Sequential([
                tf.keras.layers.Embedding(self.vocab_size,
self.embedding_dim, input_length=self.max_length),
                # tf.keras.layers.Flatten(),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128,
return_sequences=True)),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,
return_sequences=True)),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32,
return_sequences=True)),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
                tf.keras.layers.Dense(64, activation='relu'),
                tf.keras.layers.Dense(1, activation='sigmoid')
            ])

        elif type == 3:
            model = tf.keras.Sequential([
                tf.keras.layers.Embedding(self.vocab_size,
self.embedding_dim, input_length=self.max_length),
                # tf.keras.layers.Flatten(),
                tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128)),
                #tf.keras.layers.Dense(64, activation='relu'),
                tf.keras.layers.Dense(1, activation='sigmoid')
            ])

        elif type == 4:
            model = tf.keras.Sequential([
                tf.keras.layers.Embedding(self.vocab_size,
self.embedding_dim, input_length=self.max_length),
                tf.keras.layers.Conv1D(128, 5, activation='relu'),
                tf.keras.layers.GlobalAveragePooling1D(),
                tf.keras.layers.Dense(6, activation='relu'),
                tf.keras.layers.Dense(1, activation='sigmoid')
            ])

        model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
        model.summary()

        return model


    def plot_graphs(self, history, string):
        import matplotlib.pyplot as plt
        plt.plot(history.history[string])
        plt.plot(history.history['val_' + string])
        plt.xlabel("Epochs")
        plt.ylabel(string)
        plt.legend([string, 'val_' + string])
        plt.show()
```

```
    def save_graphs(self, history, string, name):
        import matplotlib.pyplot as plt
        plt.plot(history.history[string])
        plt.plot(history.history['val_' + string])
        plt.xlabel("Epochs")
        plt.ylabel(string)
        plt.legend([string, 'val_' + string])
        plt.savefig(name)
        plt.clf()



    def train_model(self, model, num_epochs=10, show_graphs="false",
save_graphs="false", name = ""):
        history = model.fit(self.training['padded'], self.training['labels'],
epochs=num_epochs,
                            validation_data=(self.validation['padded'],
self.validation['labels']))

        if show_graphs == "true":
            self.plot_graphs(history, "accuracy")
            self.plot_graphs(history, "loss")

        if save_graphs == "true":
            self.save_graphs(history, "accuracy", "01_" + name)
            self.save_graphs(history, "loss", "02_" + name)


        return model

    def predict(self, article):
        import tensorflow as tf
        from tensorflow.keras.preprocessing.sequence import pad_sequences

        from pathlib import Path
        path = Path("./classification_model/model.h5")
        path = path.resolve()
        cur = Path(".")
        cur = cur.resolve()

        model = tf.keras.models.load_model(path)
        prediction_sequences = self.tokenizer.texts_to_sequences([article])
        prediction_padded = pad_sequences(prediction_sequences,
maxlen=self.max_length, truncating=self.trunc_type)

        return model.predict(prediction_padded)[0][0]

    def evaluate_by_n_iterations(self, n, model_type, num_epochs=10,
show_graphs="false", save_graphs="false", name = ""):
        val_accuracy = []
        for x in range(n):
            print("Iteration {}:".format(x))
            model = self.create_model(type=model_type)
            model = self.train_model(model, num_epochs, show_graphs,
save_graphs, name)
            evaluated = model.evaluate(self.validation["padded"],
self.validation["labels"])
            val_accuracy.append(evaluated[1])
            del model
            print()

        print("Validation accuracy for 50 train cases:")
        for x in val_accuracy:
```

```
            print(x)

        def get_average(lst):
            return sum(lst) / len(lst)

        print()
        print("Average Validation accuracy for {} train cases:".format(n))
        print(get_average(val_accuracy))

        return get_average(val_accuracy)

def main():

    cf1 = Classifier("create", type=0)
    cf1 = cf1.train_model(cf1.model, num_epochs=15, save_graphs="true",
name="t0_15e_rm-false")
    cf1.save('model/t0_15e_rm-false.h5')

    cf = Classifier("create", type=0, rm_stopwords=True)
    cf = cf.train_model(cf.model, num_epochs=15, save_graphs = "true", name =
"t0_15e_rm-true")
    cf.save('model/t0_15e_rm-true.h5')

    cf2 = Classifier("create", type=1)
    cf2 = cf2.train_model(cf2.model, num_epochs=15, save_graphs="true",
name="t1_15e_rm-false_v2")
    cf2.save('model/t1_15e_rm-false_v2.h5')

    cf3 = Classifier("create", type=3)
    cf3 = cf3.train_model(cf3.model, num_epochs=15, save_graphs="true",
name="t3_15e_rm-false")
    cf3.save('model/t3_15e_rm-false.h5')

    cf4 = Classifier("create", type=4)
    cf4 = cf4.train_model(cf4.model, num_epochs=15, save_graphs="true",
name="t4_15e_rm-false")
    cf4.save('model/t4_15e_rm-false.h5')

    #import tensorflow as tf

    #sess =
tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(log_device_placement=Tru
e))
    #print("Num GPUs Available: ",
len(tf.config.experimental.list_physical_devices('GPU')))

    #cf.evaluate_by_n_iterations(50)

if __name__ == "__main__":
    main()
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

ВЕБ-ДОДАТОК ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ФІКТИВНИХ ТЕКСТОВИХ НОВИН У ІНТЕРНЕТ-ВИДАННЯХ

Виконав: Ільїн Максим Олександрович

Керівник: доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: рішення задачі точної класифікації текстових новин шляхом розроблення веб-додатку для виявлення фіктивних текстових новин у Інтернет-виданнях.

Завдання:

1. Проаналізувати існуючі програмні та алгоритмічні рішення.
2. Формування вимог до розроблюваної системи.
3. Обґрунтувати вибір засобів для реалізації ПЗ.
4. Розробити алгоритм класифікації та реалізувати клієнтську частину системи.
5. Протестувати розробку.

АКТУАЛЬНІСТЬ

Рішення-аналоги:

- ручна перевірка

FactCheck.org

POLITIFACT
WINNER OF THE PULITZER PRIZE

- автоматизована перевірка

SOLAT IN THE SWEN

UCL

AIPHES
ADAPTIVE INFORMATIONSAUFBEREITUNG AUS HETEROGENEN QUELLEN

Недоліки аналогів:

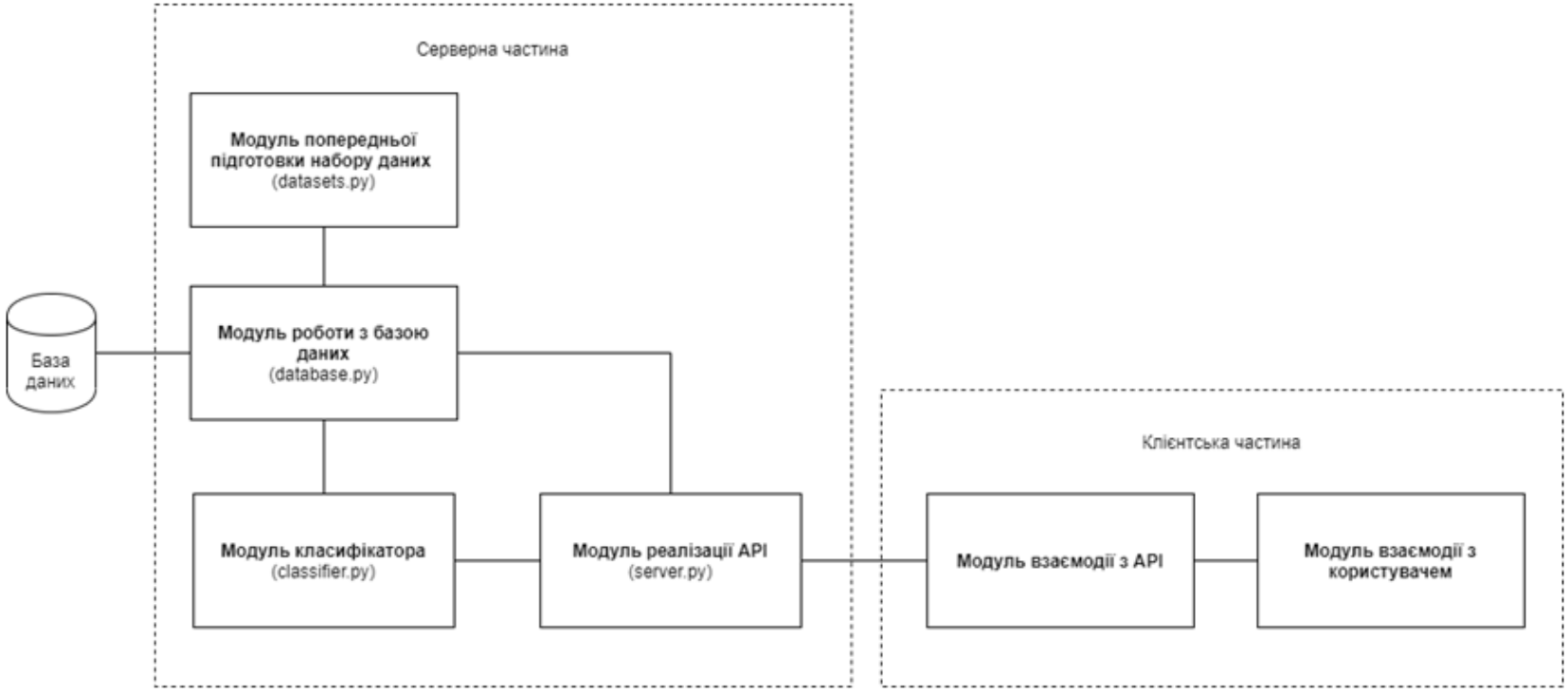
- ручна перевірка

1. Низька швидкість перевірки.
2. Висока витрата людського ресурсу.

- автоматизована перевірка

1. Відсутність графічного інтерфейсу користувача.
2. Низький рівень абстракції (неможливість використання не-програмістом).
3. Залежності точності класифікації від набору даних, на якому навчалась модель (50-99%).

Загальна структура системи



ЗАСОБИ РЕАЛІЗАЦІЇ

HTML

JS

CSS

Flask
web development,
one drop at a time

TensorFlow

mongoDB

gunicorn

НАБІР ДАНИХ

Fake and real news dataset

+

FakeNewsNet

< **Fake.csv** (59.88 MB)

Detail Compact Column

title	text	subject	date
The title of the article	The text of the article	The subject of the article	The date at which the article was posted

```
├── gossipcop
│   ├── fake
│   │   ├── gossipcop-887096424105627648
│   │   │       └── news content.json
│   │   └── ....
│   └── real
│       ├── gossipcop-887096424105627652
│       │       └── news content.json
│       └── ....
├── politifact
│   ├── fake
│   │   ├── politifact-554489424105627648
│   │   │       └── news content.json
│   │   └── ....
│   └── real
│       ├── politifact-735146424105627652
│       │       └── news content.json
│       └── ....
```

51 237 маркованих новин

АНАЛІЗ НАБОРУ ДАНИХ

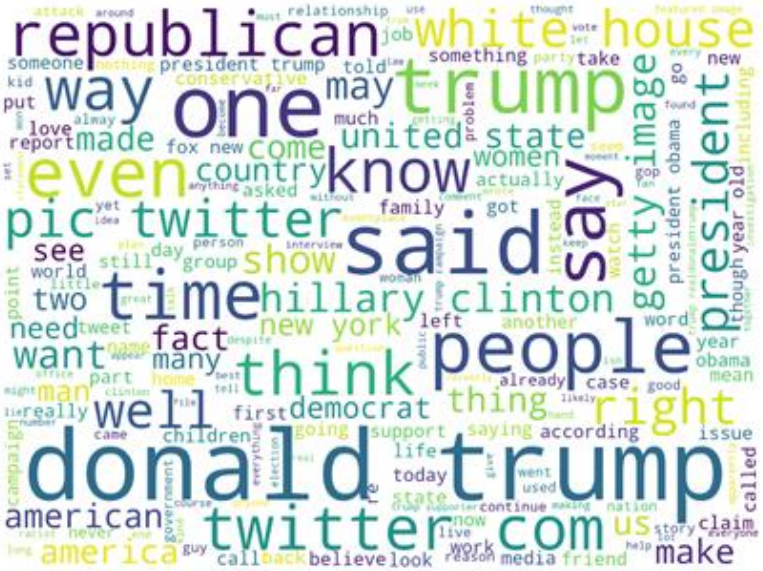
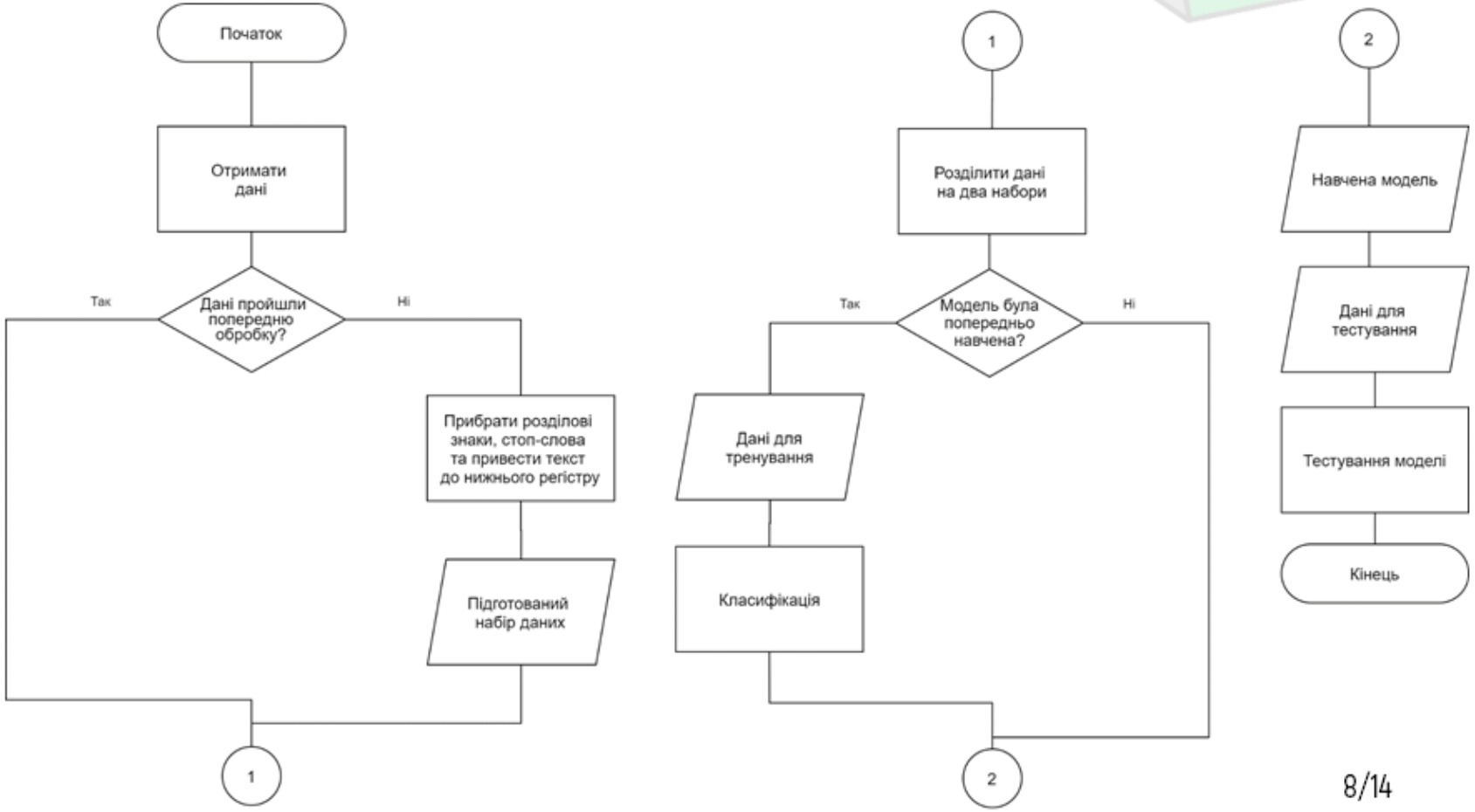


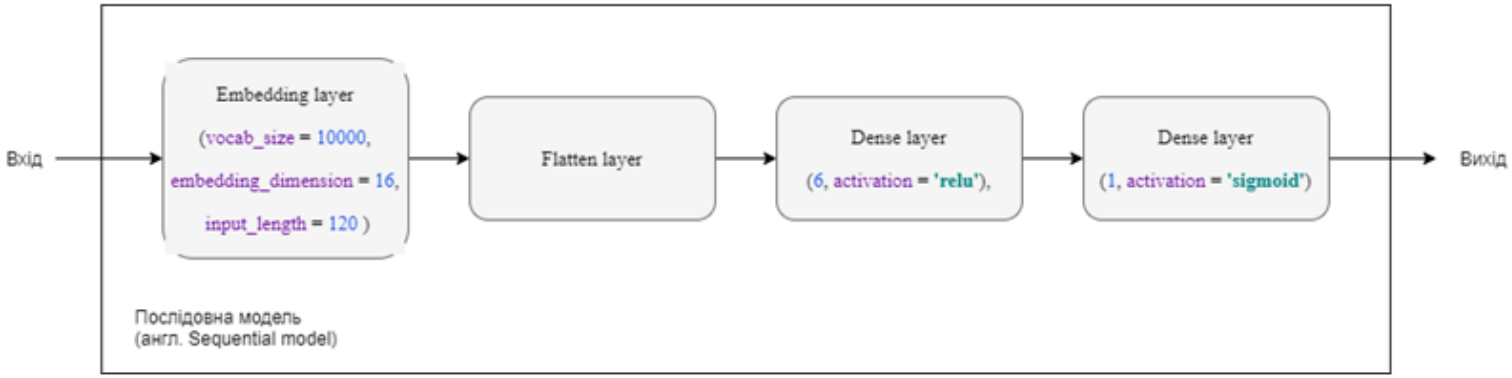
Рис. 1 . Представлений візуально зважений список слів, що зустрічаються у тексті фіктивних новин

Рис. 2. Представлений візуально зважений список слів, що зустрічаються у тексті правдивих новин

РЕАЛІЗОВАНИЙ АЛГОРИТМ КЛАСИФІКАЦІЇ



МОДЕЛЬ КЛАСИФІКАТОРА



РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Назва розробки	Точність класифікації (%)
Talos Intelligence	82.02 %
athene_system	81.97 %
UCL Machine Reading	81,72%
Fake or Real	93,15 %

КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС

Рис. 1 . Сторінка вводу тексту новини

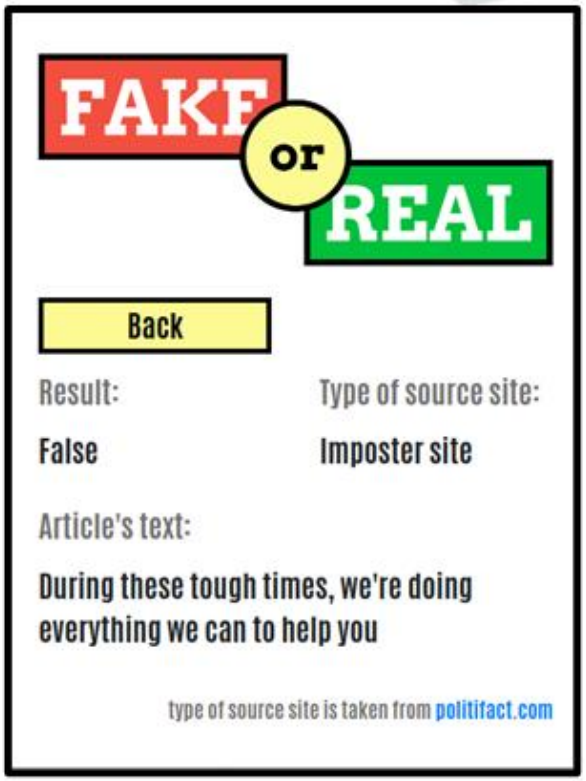


Рис. 2 . Сторінка отримання результату аналізу

АПРОБАЦІЯ

Даний дипломний проєкт був представлений на Міжнародній науковій конференції «Інтелектуальні системи прийняття рішень та проблеми комп'ютерного інтелекту» (ISDMCI'2020) і опублікований у вигляді тез доповіді.

Підготовано пакет документів на отримання авторського свідоцтва.

ВИСНОВКИ

1. Проаналізовано проблематику класифікації текстових новин та існуючі програмні рішення.
2. Обґрунтовано вибір програмних засобів розроблення.
3. Розроблено веб-додаток, що виконує класифікацію користувацької текстової новини як фіктивної або правдивої алгоритмом машинного навчання.
4. Виконано мануальне тестування розробленого веб-додатку.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ВЕБ-ДОДАТОК ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ФІКТИВНИХ ТЕКСТОВИХ НОВИН У ІНТЕРНЕТ-ВИДАННЯХ

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Максим ІЛЬЇН

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях, який представлено у вигляді розширення для веб-браузеру Google Chrome.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Можливість коректного налаштування розширення веб-браузера.
2. Коректність та стабільність користувацької взаємодії з розширенням.
3. Відповідність клієнтської частини нефункціональним вимогам (інтуїтивна зрозумілість інтерфейсу, зручність використання тощо).
4. Відповідність серверної частини нефункціональним вимогам (час відгуку, обсяг тексту, який може бути коректно проаналізований розробленими засобами тощо).

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується відповідно до техніки Smoke Testing, в ході якої відбувається перевірка основної функціональності системи на базовому рівні.

Використовуються наступні методи:

1. Функціональне тестування, зокрема на рівні GUI testing (перевірка користувацької взаємодії).
2. Нефункціональне тестування, зокрема на рівні Reability testing (перевірка надійності системи) та Efficienty testing (перевірка ефективності системи).

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність програмного продукту перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних значень в текстові поля, що є доступними для використання у рамках системи.
2. Статичного тестування коду.
3. Динамічного ручного тестування на відповідність функціональним вимогам.
4. Тестування користувацького інтерфейсу.
5. Тестування програмного застосунку у веб-браузері Google Chrome.
6. Тестування зручності використання.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2020 р.

ВЕБ-ДОДАТОК ДЛЯ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ФІКТИВНИХ ТЕКСТОВИХ НОВИН У ІНТЕРНЕТ-ВИДАННЯХ

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Максим ІЛЬЇН

2020

ЗМІСТ

1. Опис процесу встановлення програмного застосунку

Веб-додаток для автоматизованого виявлення фіктивних текстових новин у Інтернет-виданнях являє собою розширення для веб-браузера Google Chrome, що встановлюється користувачем через відповідне меню «Розширення» браузера. Після встановлення піктограма розширення з'являється у верхній навігаційній панелі браузера серед інших іконок розширень веб-браузера.

2. Опис процедури налаштування системи для роботи

Після встановлення, розширення є повною мірою готовим до роботи.

Процес налаштування веб-додатку для автоматизованого виявлення фіктивних текстових новин в браузері полягає в активації системи натисненням відповідної кнопки у правому верхньому куті навігаційної панелі веб-браузера (рис. 2.1).

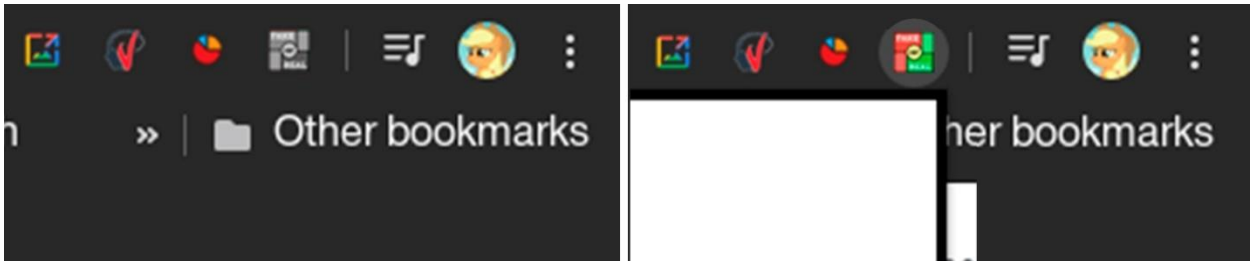


Рис. 2.1. Ілюстрація процесу активації розширення

Активоване користувацьке розширення використовує текстовий вміст активної вкладки браузера, не змінюючи його, у вкладці із користувацьким інтерфейсом розширення, що забезпечує взаємодію із клієнтською частиною системи.

3. Користування програмними засобами

Функціональність розширення дозволяє копіювати виділений текст з будь-якої веб-сторінки, що переглядає користувач, тому для початку його

використання після завантаження сторінки сайту Інтернет-видання не треба виконувати жодних додаткових дій.

Виділивши блок тексту, що його цікавить, користувач може активувати розширення, та натиснути на кнопку “Paste selection” (рис. 3.1), що скопіює виділений текст у спеціалізоване поле вводу. Копіювання тексту також можна здійснити з довільного джерела засобами ОС. Також підтримується редагування скопійованого тексту. Після виділення тексту, для виконання аналізу потрібно натиснути кнопку “Check”.

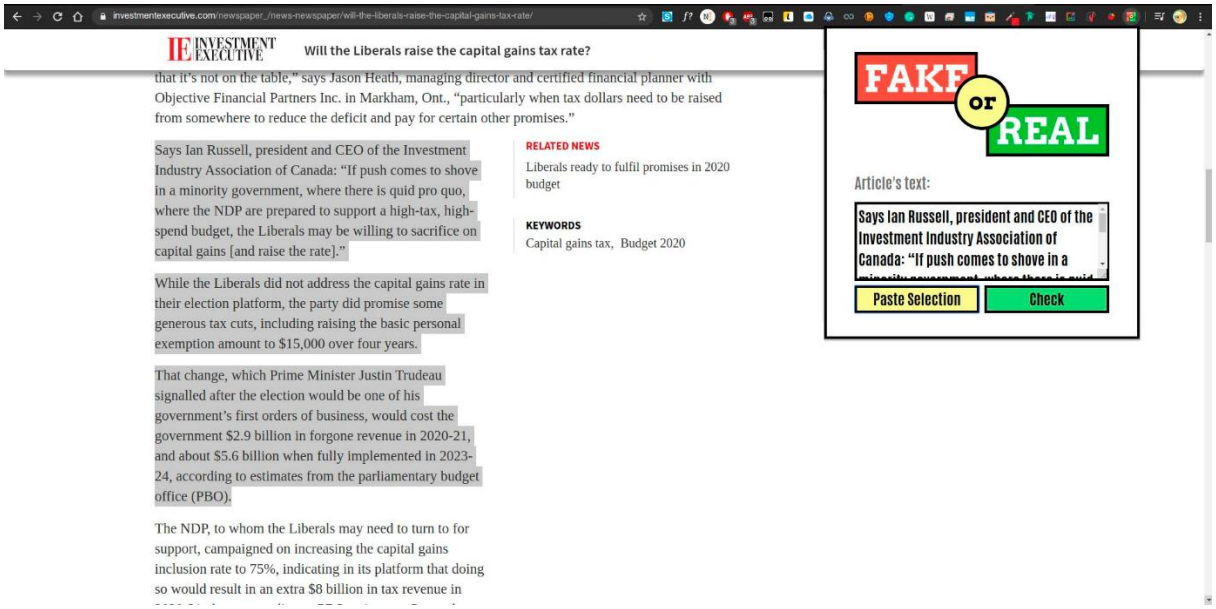


Рис. 3.1. Ілюстрація вставки виділеного тексту

Результат аналізу буде відображено на новій сторінці під написом “Result”. Також під написом “Type of source site” користувач може переглянути рейтинг сайту-джерела новини відповідно до правдивості новин, що публікує видання (рис. 3.2).

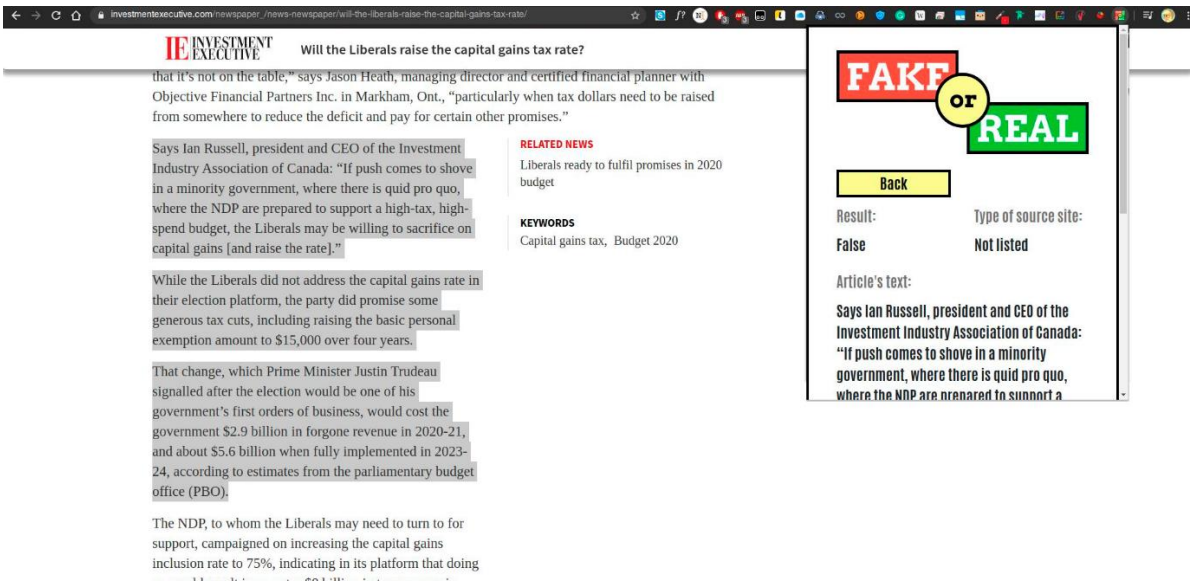


Рис. 3.2. Результат аналізу тексту новини на правдивість

Після виконання аналізу користувач може вийти із розширення натиснувши у будь-якому місці поза вікном розширення, або натиснути на кнопку “Back” та перейти до попереднього екрану розширення.